Coordinated Multi-Agent Planning and Scheduling

Proefschrift

der verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties
in het openbaar te verdedigen op vrijdag 17 mei 2013 om 15:00 uur
door Jan Renze STEENHUISEN
ingenieur technische informatica
geboren te Hoogeveen
Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. C. Witteveen

Copromotor: Dr. T.B. Klos

Samenstelling promotiecommissie:

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rector Magnificus</td>
<td>voorzitter</td>
</tr>
<tr>
<td>Prof. dr. C. Witteveen</td>
<td>Technische Universiteit Delft, promotor</td>
</tr>
<tr>
<td>Dr. T.B. Klos</td>
<td>Technische Universiteit Delft, copromotor</td>
</tr>
<tr>
<td>Prof. dr. S.F. Smith</td>
<td>Carnegie Mellon University, Verenigde Staten van Amerika</td>
</tr>
<tr>
<td>Prof. dr. T.C.M. Holvoet</td>
<td>Katholieke Universiteit Leuven, België</td>
</tr>
<tr>
<td>Prof. dr. J-J.Ch. Meyer</td>
<td>Universiteit Utrecht</td>
</tr>
<tr>
<td>Prof. dr. ir. J.A. La Poutré</td>
<td>Universiteit Utrecht/ Centrum Wiskunde &amp; Informatica</td>
</tr>
<tr>
<td>Prof. dr. K.L.M. Bertels</td>
<td>Technische Universiteit Delft</td>
</tr>
<tr>
<td>Prof. ir. dr. D.H.J. Epema</td>
<td>Technische Universiteit Eindhoven/ Technische Universiteit Delft, reservelid</td>
</tr>
</tbody>
</table>

SIKS
Dissertation Series No. 2013-19

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems, and supported by the ICIS (Interactive Collaborative Information Systems) project, funded by the BSIK program of the Dutch Ministry of Economic Affairs, grant number BSIK03024.

© 2013 by J.R. Steenhuisen
Printed by Gildeprint, Enschede

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilised in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without the prior permission of the author.
# Contents

1 Introduction .................................................. 1
   1.1 Example Scenarios ................................... 4
      1.1.1 Aircraft Ground Handling ....................... 4
      1.1.2 Port Container Transshipment ................. 8
   1.2 Coordinated Planning and Scheduling .............. 10
   1.3 Contributions and Thesis Outline .................. 14

2 Coordinated Planning and Scheduling .................. 17
   2.1 Multi-Agent Task-Based Planning and Scheduling .... 18
      2.1.1 Multi-Agent Task-Based Planning .............. 19
      2.1.2 Multi-Agent Task-Based Scheduling .......... 24
   2.2 Coordinating Agents’ Plans and Schedules .......... 27
      2.2.1 A-Posteriori Coordination ..................... 28
      2.2.2 Interleaved Coordination ....................... 29
      2.2.3 A-Priori Coordination ......................... 30
   2.3 A-Priori Coordinated Planning and Scheduling .... 32
      2.3.1 Plan Coordination .................................. 33
      2.3.2 Temporal Decoupling ............................ 35
   2.4 Discussion ................................................ 37
   2.5 Research Questions and Thesis Outline ............. 40

3 Decoupling Qualitative Task Networks ................. 43
   3.1 Qualitative Task Networks ......................... 44
      3.1.1 Agents and Task Assignments .................. 45
      3.1.2 Agent Plans ....................................... 46
      3.1.3 Plan Coordination ................................ 47
   3.2 Plan Decoupling ....................................... 50
      3.2.1 Complexity ......................................... 54
      3.2.2 Factors Influencing the Complexity of PLAN-DECOUPLING PROBLEMS ......................... 55
      3.2.3 A Heuristic for Constructing Coordination Sets ............... 59
   3.3 Discussion ............................................... 63
## 4 Decoupling Quantitative Task Networks

4.1 Quantitative Task Networks .................................................. 67
   4.1.1 Plans in STTNs .......................................................... 69
   4.1.2 Plan Coordination ....................................................... 71
4.2 Plan Decoupling in STTNs .................................................. 72
4.3 Schedule Coordination .......................................................... 75
4.4 Dispatch Coordination .......................................................... 77
4.5 Relation Plan and Schedule Coordination .................................. 79
4.6 Discussion ............................................................................. 80

## 5 Empirical Evaluation

5.1 System Perspective ................................................................. 84
   5.1.1 Expectations ..................................................................... 85
   5.1.2 Design of Experiments .................................................... 86
   5.1.3 Experimental Setup .......................................................... 87
5.2 System Perspective ................................................................. 90
   5.2.1 The effect of plan decoupling on plan-execution cost .......... 90
   5.2.2 The effect of plan decoupling on plan-construction time ....... 93
5.3 Agent Perspective ................................................................. 98
   5.3.1 Expectations ..................................................................... 98
   5.3.2 Design of Experiments .................................................... 99
   5.3.3 Experimental Setup .......................................................... 100
5.4 Agent Perspective ................................................................. 101
   5.4.1 The effect of plan decoupling on plan-execution cost .......... 101
   5.4.2 The effect of plan decoupling on plan-construction time ....... 104
5.5 Summary and Conclusion ........................................................ 105

## 6 Conclusions and Research Directions

6.1 Answer to Research Questions ................................................ 107

## A Planning Preliminaries

A.1 Planning Problems ................................................................. 115
A.2 Planners and Plan Costs .......................................................... 118

## B Materials and Methods

B.1 Planning Problems ................................................................. 121
   B.1.1 Logistics Problem ............................................................ 121
   B.1.2 Depot Problem ................................................................. 122
B.2 Task Network Representation .................................................. 124
   B.2.1 Logistics Problem ............................................................ 124
   B.2.2 Depot Problem ................................................................. 125
B.3 Plan decoupling ................................................................. 126
   B.3.1 Logistics Problem ............................................................ 126
   B.3.2 Depot Problem ................................................................. 127
Chapter 1

Introduction

Computer systems have been revolutionising our world ever since they have come into existence, both at work and in our private lives. Nowadays, some people even consider the Internet as a necessity of life. Many younger people can’t even imagine a world without e-mail and instant messaging, although these services have come into existence only recently. These and similar developments daily progress and are likely to keep changing our way of thinking, communicating and living in the many years to come.

Also in business processes, computer systems have become of increasing importance. Most—if not all—companies use computers for administrative purposes, and many stores offer their goods online, often using a shipping service for home delivery. Other businesses offer services, ranging from non-automated (e.g., chess tutoring) via semi-automated (e.g., stock-trading advice) to fully-automated services (e.g., search engines). Also for these businesses, a tendency can be observed of moving towards offering automated services online. Moreover, we see organisations being structured as sets of services, such as universities where persons, departments, and physical assets are interconnected by a computer network.

With computer systems becoming more and more sophisticated, it does not take much imagination to realise that at some time we will have computer systems that are acting and providing composite services by themselves. These systems, called agents, must be able to interact and cooperate with other systems, while guaranteeing that their joint behaviour is desirable. Currently, we already see developments in this direction with agents given goals of increasing complexity and importance to pursue (e.g., air-traffic control (Léauté and Williams, 2005) and disaster rescue (Harrauld, 2006)).

In order to achieve their goals, such agents need to make all kinds of decisions. It is common to make a distinction between three levels of decision making: strategic, tactical, and operational. There, decision problems need to be solved, which requires increasing levels of temporal detail.

On the strategic level, planning problems need to be solved that specify how to achieve a certain goal, called the plan, while leaving out unnecessary (temporal) details. For example, assume a company wants to sell motorcycles in various locations
across Europe. The company needs to decide where to produce them and how they will be transported to the European market. In partial fulfilment of this goal, it can be decided to use a factory based in Chongqing, China that has a maximum production capacity of 36 motorcycles weekly, and to transport them by ship to North-West Europe. Additionally, a plan for transportation is needed which requires a seaport in China (e.g., Guangzhou, or Hong Kong) and a seaport in North-West Europe (e.g., Rotterdam, or Antwerp) to be chosen, and how transportation will take place to and from these harbours (e.g., by train or truck). The plan is constructed to transport the motorcycles by truck to Guangzhou, where they are shipped onto a container ship which transports them to the Port of Rotterdam, from where they are loaded onto trucks for delivery at their destinations. In addition, three truck rides are made available for each week, and a contract is signed with a shipping company that provides a monthly container-shipping service between Guangzhou and Rotterdam.

On the tactical level, scheduling problems need to be solved that also result in plans but then with temporal information (e.g., setting task deadlines and task durations). For solving these problems, some choices need to be made beforehand, such as the number of motorcycles to produce. In constructing such a schedule, the focus is on cost-related issues, such as resource allocation (e.g., number of machines), resource utilisation, and robustness against operational disruptions. Continuing with the example, it is decided to produce 24 motorcycles and to transport them in two sea containers each week. In Guangzhou, the containers are stacked, until they can be loaded onto a container ship that leaves every first Monday of the month for a journey of approximately four weeks to the Port of Rotterdam.

On the operational level, the problem is to keep the execution of actions according to the constructed schedule, specifying exactly when the tasks must be executed (i.e., a dispatching). Given the factory’s production capacity, it is decided to produce four motorcycles every working day which requires all kinds of additional decisions (e.g., staffing). A sea container filled with 12 motorcycles is then loaded onto a truck on Wednesday and Saturday evenings and is driven to the harbour where it arrives two days later in the morning. Then, on the first Monday of the month, the container ship is loaded with the stacked containers together with the morning arrival, and leaves in the afternoon for Rotterdam.

Although it can be debated which problems need to be solved in which of these three levels of decision making, the types of problems that need to be solved are not disputed. Therefore, we will use the distinction between a planning, scheduling,
and dispatching phase as shown in Figure 1.1. Here, *planning* involves selecting and ordering actions that achieve a goal using *qualitative*-temporal constraints, while *scheduling* adds *quantitative*-temporal constraints (e.g., deadlines) for completing the plan (Ghallab et al., 2004). Then, execution needs to take place according to the schedule (i.e., within the constraints of the schedule). This requires a *dispatching* to be constructed for the schedule, which is a specification of the exact starting times of all tasks that satisfies all planning and scheduling constraints. The results of both the planning and scheduling phase need to be such that a dispatching is guaranteed to exist (i.e., that the constructed plans and schedules can be executed).

Although these problems are already difficult when considering a single agent, new problems arise when dealing with multiple agents. A *multi-agent system* (MAS) consists of a set of agents, each with its own capabilities (Weiß, 1999; Wooldridge, 2002).

*Autonomy* is considered a distinctive agent property (Jennings, 1996; Wooldridge and Jennings, 1995), which needs to be adjustable such that the agents are able to take up roles ranging from being obedient to requiring self-interested (i.e., fully-autonomous) behaviour (Barber and Martin, 2002; Sellner et al., 2006; van der Vecht et al., 2006). Obedient agents have no autonomy, which means that even in the dispatching phase no decision is made by themselves. Self-interested agents, on the other hand, want to make decisions in all of the above mentioned phases. It is necessary that the agents are guaranteed to exhibit desired behaviour at every level of autonomy. We consider this desired behaviour to involve achieving a (common) goal which is specified as a *task network* (i.e., a set of tasks together with some constraints) that has to be completed.

Agents might have a common planning/scheduling goal which cannot be achieved by any single agent alone. For example, in multi-modal logistics, transporting a package might require both a truck and a ship. Here, agents need to cooperate while remaining autonomous to some extent. When multiple agents need to work together to achieve such a goal, they become *dependent* on each other. These dependencies must be taken into account in the decision making of the planning, scheduling, and dispatching phase.

The central problem then becomes how to reconcile the autonomy of the agents with satisfying the task dependencies. This thesis considers MASs that have to achieve a goal, specified as the completion of a task network (i.e., a constrained set of tasks), which requires a set of services that are distributed over the agents. Because each agent is able to offer only a limited number of services, multiple agents need to cooperate in constructing plans and schedules for the services of each of the agents to achieve the goal. Self-interested agents, however, want to solve their own local planning and scheduling problems independently from the other agents. As we will show, independently-constructed plans and schedules together might be inconsistent (i.e., that a dispatching might not exists that satisfies all planning or scheduling constraints), even when all these independently-constructed plans and schedules are *locally consistent* (i.e., a dispatching exists that satisfies all local planning or scheduling constraints).
The goal of this thesis is to enable a set of autonomous agents to construct a global solution for a planning or scheduling problem by coordinating among the agents such that the independently constructed locally-consistent plans or schedules together form such a global plan or schedule.

While coordination problems can be solved for agents at different levels of autonomy, the most challenging coordination problems are those for self-interested agents. Because these agents maintain the highest level of autonomy, they pose the hardest constraints on solving the coordination problems. When a coordination mechanism is acceptable for self-interested agents, then it is—from an autonomy point of view—acceptable for agents at any level of autonomy. Therefore, in this thesis, we focus on the most challenging type of agents, self-interested agents.

1.1 Example Scenarios

To illustrate the concepts and ideas of the previous section, we describe two example scenarios in which multiple self-interested agents together provide a certain service. Although we describe only two scenarios, multi-agent systems are studied in a wide variety of domains. Application domains include multi-modal transportation (Aronson et al., 2002; Chiu et al., 2005), hospital patient scheduling (Paulussen et al., 2003; Zöller et al., 2006), search and rescue (Koes et al., 2006), aircraft ground handling (Neiman and Lesser, 1996; van Leeuwen et al., 2007), space exploration (Sellner et al., 2006), and crisis response (Harald, 2006; van Veelen et al., 2008; Wagner et al., 2004). The goal of this section is to get some feeling for the kind of problems we have in mind, and some intuition on the coordination problems that can arise.

1.1.1 Aircraft Ground Handling

Aircraft ground handling involves the servicing of an aircraft while it is on the ground and (usually) parked at a terminal gate of an airport. Ground handling includes the many service requirements of a passenger aircraft between the time it arrives at a terminal gate and the time it departs for its next flight. In Figure 1.2, two such aircraft ground-handling services are depicted as examples. Speed, efficiency, and accuracy are important in order to minimise this turnaround time. Some important ground-handling services are cabin cleaning, catering, refuelling, lavatory drainage, luggage (un)loading, and staffing of the different counters for departure and arrival. When such a service is needed, it can be considered a task which needs to be completed.

Example 1.1. Consider the cabin-cleaning service, which involves both cleaning and replenishment of toiletries, pillows, and blankets. Providing this service for a certain plane can be described as multiple tasks that need to be completed. For instance, we could define a task for the cleaning of the cabin, and three tasks for the replenishment of toiletries, pillows, and blankets. If desirable, finer-grained tasks...
1.1. EXAMPLE SCENARIOS

(a) A tug pulling a plane (Adrian Pingstone).

(b) Lavatory drainage.

Figure 1.2: Impressions of aircraft ground handling (source: Wikipedia).

can be defined to describe the cleaning of different sections of the cabin, and the cleaning and replenishment of toiletries for the different toilets separately.

For each task, an agent is needed that is capable of completing it (i.e., offers the required service) during the time that the plane is parked on the ground. Each of the ground-handling services could be offered by multiple agents, and each agent might be able to offer multiple services. Instead of finding an agent for each single service each time, most airlines have contracts with specific companies (i.e., agents) for providing the required services. Therefore, for each plane of some airline, the assignment of tasks to the service-providing agents is determined by these contracts.

Let us now focus on a specific airport. Amsterdam Airport Schiphol is the Netherlands’ main international airport and an important European hub. In 2011, with a total of 49,680,625 passengers handled, it ranked as Europe’s 4th busiest by overall passenger volume (CBSstatline, 2012). In the same year, Schiphol handled 1,523,803 tons of cargo, ranking it as Europe’s 4th largest in cargo tonnage.

The control tower is responsible for constructing plans and schedules for the safe arrival and departure of each plane. These assigned time windows for arrival and departure of each plane constrain the total execution time of providing all ground-handling services together.

Example 1.2. Consider the situation of a plane for which the control tower has decided that it is allowed to land at 13:40 and may take off again at 15:20. Because it has been assigned to the Polderbaan, which is the runway farthest away from the terminals, for both landing and take off, taxiing takes exactly 20 minutes single ride. Therefore, all ground-handling tasks have to be completed in the time window [14:00, 15:00].
At Schiphol, most ground-handling services are offered by multiple agents. For instance, there are 3 fuelling companies and 6 companies that offer catering services (i.e., the unloading of unused and loading of fresh food). For each arriving plane, the required ground-handling services are assigned as tasks to the agents. Despite the fact that the agents represent independent companies or organisations, they become dependent on each other when they are assigned tasks for servicing the same plane. All these agents need to construct plans and schedules for executing their tasks, such that the collective ground-handling service (i.e., all services together) is provided. These plans and schedules need to satisfy some constraints. First, they have to be in compliance with the time constraints set by the control tower. Second, additional constraints need to be met which are due to system specifications (e.g., the speed at which refuelling can take place), or safety regulations (e.g., the minimum amount of fuel for the plane’s next flight).

**Example 1.3.** Consider the task of refuelling (R) a plane, and the deboarding (D) and boarding (B) of passengers. Due to safety regulations, it is not allowed to do any refuelling whilst there are passengers on board the plane. Therefore, these tasks must be executed in the following order: \( D \) before \( R \), \( R \) before \( B \).

On average, approximately 600 planes arrive and depart at Schiphol on one day. Before a plane can depart, it requires some ground-handling services (e.g., refuelling). The types of services depend on the plane and type of flight (e.g., passenger or cargo).

In this ground-handling scenario, we have seen that a vast number of tasks needs to be completed every day. Moreover, execution of these tasks is constrained in a number of ways (e.g., execution intervals and ordering constraints). Although the agents represent independent companies, they become dependent when they are involved in servicing the same plane. Therefore, carefully-constructed plans and schedules are needed such that these dependencies are satisfied.

In the following example, we show how conflicts might arise when agents independently construct schedules.

**Example 1.4.** Consider the plane arriving at 14:00 and departing at 15:00, and that it requires refuelling (R) and to change passengers, which involves deboarding (D) and boarding (B). As described in Example 1.3, the following ordering constraints need to be satisfied for safety reasons: \( D \) before \( R \) and \( R \) before \( B \). Moreover, deboarding and boarding will both last between 10 and 15 minutes, while the refuelling task requires 10 to 20 minutes for completion.

There are two agents that are responsible for the (de)boarding and refuelling task, respectively, that need to be completed in the time window \([14:00, 15:00]\). While the (de)boarding agent needs to schedule its tasks \( D, B \) in such a way that \( D \) before \( B \) is satisfied, the refuelling agent does not have any local ordering constraints on its task \( R \). Note that none of the agents is responsible for satisfying the ordering constraints \( D \) before \( R \) and \( R \) before \( B \).

Now consider the schedules as depicted in Figures 1.3(a) and 1.3(b). In both situations, the (de)boarding agent has scheduled its deboarding task to be executed...
between 14:10 and 14:25, and boarding to be executed between 14:35 and 14:50 (both tasks have a duration of at most 15 minutes). Clearly, this schedule is locally consistent because it allows the dispatching of starting the deboarding task at 14:10 and the boarding task at 14:35, which satisfies both the ordering constraint D before B and the quantitative constraints (i.e., both scheduling constraints and the time window [14:00, 15:00]). The refuelling agent has scheduled its task (i) between 14:20 and 14:35 (in Figure 1.3(a)), and (ii) between 14:25 and 14:35 (in Figure 1.3(b)), respectively. Again, these schedules are locally consistent because they allow dispatchings where the refuelling task is started at the earliest starting time (i.e., 14:20 or 14:25), which leaves (just) enough time to complete the task (i.e., 15 and 10 minutes, respectively). These dispatchings satisfy all scheduling constraints (i.e., the tasks are started not earlier than their earliest starting times, and can be completed not later than the latest ending time) and the time window [14:00, 15:00].

In order to form a schedule for the complete ground-handling task, these locally-consistent schedules together also need to satisfy the remaining inter-agent constraints: D before R and R before B. While the latter constraint is satisfied in both situations (i.e., refuelling is completed before boarding starts), deboarding is not scheduled to be completed before starting refuelling in both situations. In Figure 1.3(a), the constraint D before R is not satisfied when deboarding (D) is scheduled to be finished at 14:25, and refuelling (R) to be started at 14:20 (i.e., 14:25 ≠ 14:20).

In this example, it was shown that a scheduling conflict can arise when agents independently construct their schedules. Although all constructed schedules were locally consistent, these schedules together did not satisfy an inter-agent constraint. Therefore, these inter-agent dependencies necessitate some management to take place, because they can cause conflicts when the agents schedule independently. We can prevent these conflicts by changing the agents’ local constraints such that the inter-agent dependencies are always satisfied.
1.1.2 Port Container Transshipment

The main service provided by a sea port is that of transshipment of bulk goods and containers. Transshipment is a means to increase transport efficiency and involves two main activities. First, we have inter-modal transshipment, which involves the changing of modality (i.e., type of transport), such as from water to rail or road transport. Second, transshipment can be used to combine multiple smaller shipments into one large shipment (e.g., from multiple river barges into a sea vessel) or vice versa.

For this example scenario, let us focus on the transshipment of containers at a container terminal. At such a terminal, different types of container cranes (e.g., the waterside cranes) are used for the transshipment of containers of different modalities. Irrespective of the type of transshipment, however, each container is unloaded off a container ship and transported to the stack. There, it is temporarily stored until it can be loaded onto its outgoing means of transport. The total container flow handled at a terminal is commonly expressed in twenty-foot equivalent units (TEU), because containers of different sizes exist. Most container cranes, or hoists, are capable of lifting only one container at a time, irrespective of its size measured in TEU. Therefore, the container flow does not necessarily reflect the number of processed containers by a terminal.

Example 1.5. Consider the Emma Mærsk container carrier arriving at Rotterdam, loaded with 10,000 TEU containers. At the terminal, where it will dock, four cranes need to unload all containers. Two freight trains are already waiting for 200 TEU of these arriving containers for their first transport, and five river barges are waiting for another 2,000 TEU. Here, an unloading task exists for each container and, depending on the container, it then needs to be loaded onto the waiting train or barge, or brought to the stack where it can be stored. Clearly, ordering constraints exist between the tasks that are concerned with each single container (e.g., it must be unloaded before it can be transported).

The Port of Rotterdam (see Figure 1.4 for some impressions) is the largest port in Europe.\(^1\) Covering 105 square kilometres, the Port of Rotterdam stretches over a distance of 40 kilometres, from Delfshaven (i.e., the city centre’s historic harbour area) to the reclaimed Maasvlakte. Furthermore, the Maasvlakte 2 is under construction, and is expected to service the first ships in 2013. From 1962 until 2004 Rotterdam was the world’s busiest port based on the annual cargo tonnage, but is now overtaken by Asian ports.

Transshipment of containers takes place among sea vessels, river barges, trains, and road trucks. The port is located in the Meuse and Rhine river delta at the North Sea shore, making it well reachable for large sea vessels. For barges, these rivers provide excellent access to the hinterland, and the Betuweroute provides a fast cargo railway connection into Germany.

\(^1\)Until 2004, Rotterdam was also the world’s largest port.
1.1. EXAMPLE SCENARIOS

Figure 1.4: Impressions from the Port of Rotterdam (source: Wikipedia).

Example 1.6. Due to regulations concerning freight transport by rail, the freight train mentioned in Example 1.5 should leave before 01:00. This can be done, but requires some 1,000 containers to be unloaded first, because they are stacked on top of the 200 containers designated for the first transports by train. Therefore, for meeting the mentioned deadline, it is preferred that some ordering constraints between the unloading of some containers are satisfied (i.e., besides the ordering constraints that are due to the stacking order).

Example 1.7. Due to the availability of a quay with a draft of 24 meters, Rotterdam is one of only two ports where the world’s largest bulk carrier, Berge Stahl, can dock. This bulk carrier has a draft of just over 23 meters, which leaves only 1 meter of under keel clearance when mooring at that quay. Therefore, it can only dock in a restricted tidal window.

On average, approximately 100 sea vessels require the transshipment of almost 30,000 TEU, each day. There are 9 container terminals available for container ships equipped with 103 container cranes in total.

In this container-transshipment scenario, we have seen that a vast number of tasks needs to be completed every day. It might be preferred or even required to impose constraints on the execution order or on the execution times, due to the combination of the physical dimensions of a ship and a dock. Finally, we saw that the number of terminals, and the number of available cranes that need to complete all these tasks, is limited even at a large port. Therefore, carefully-constructed plans and schedules for all agents together are needed for efficient container transshipment.

However, the agents are mostly interested in the costs of their own plans and schedules, because they represent separate companies. Constructing such efficient local plans might result in inefficiency in the overall plan, or deadlocks even, as illustrated in the following example.
Example 1.8. Consider a container ship where a container needs to be unloaded (U) and brought to the stack (T), and replaced by a container that needs to be brought from the stack (F) and loaded onto the ship (L). These pairs of tasks are constrained by $U \text{ before } T$ and $F \text{ before } L$. Here, two agents are involved in completing the tasks: a container-crane agent and a transport agent, responsible for container transport from and to the stack. The transport agent is involved in all tasks, while the container-crane agent is only involved in the (un)loading tasks $U, L$. This leads to the task assignment as depicted in Figure 1.5.

The container-crane agent wants to complete container transshipment in the shortest period of time (i.e., from the container ship’s point of view). Therefore, it prefers a plan where unloading the container precedes the loading of the other container in its place (i.e., $U \text{ before } L$). The transport agent, on the other hand, does not want to use two trucks for these tasks, nor does it want to drive twice. Consequently, this agent prefers the plan that the container is brought from the stack to the crane that loads it onto the ship, and then unloads the container that needs to be taken to the stack (i.e., $L \text{ before } U$).

Clearly, these plans are not compatible while they result in a deadlock, when the truck can carry only one container at a time.

In this example, it was shown that independent planning can result in a deadlock which means that no dispatching exists for it that satisfies all constraints (i.e., both $U \text{ before } L$ and $L \text{ before } U$). This conflict is caused by independent local planning and does not depend on the schedules constructed by the agents. Therefore, preventing such conflicts can be achieved by restricting the local plans that the agents are allowed to construct. We can restrict the local plans by adding (ordering) constraints to the agents’ local problems.

1.2 Coordinated Planning and Scheduling

In the previous section, we showed that inter-agent conflicts can arise when all agents independently construct plans and schedules for their assigned sets of tasks, even when all local task dependencies are satisfied. These independently constructed plans and schedules together must also satisfy all dependencies between tasks that
are assigned to different agents. Because such inter-agent dependencies are not considered by any of the agents, they might not be satisfied in the joint plan or schedule. Problem instances where such inter-agent conflicts can arise are called uncoordinated.

**Example 1.9.** Continuing Example 1.4, consider a (de)boarding agent and a refuelling agent that need to service two planes of flights NW231 and NW142 between 14:00 and 15:00. For each flight, a deboarding (D), refuelling (R), and a boarding (B) task need to be completed, each requiring exactly 10 minutes, and have the following ordering dependencies: \( D \) before \( R \) before \( B \). In total, six tasks need to be completed (i.e., three for each flight), with the refuelling agent being assigned 2 tasks and the (de)boarding agent 4 tasks.

In this situation, a deadlock arises when the refuelling agent prefers completing the refuelling task of flight NW231 before that of NW142, and the (de)boarding agent prefers completing the tasks for flight NW142 before servicing NW231.

In order to resolve these inter-agent conflicts, agents need to coordinate among themselves, which involves “managing dependencies between activities” (Malone and Crowston, 1994). Considering agents to be rational, it can be assumed that they are able to manage the dependencies among their local activities. Therefore, the coordination problem is to guarantee the satisfaction of the inter-agent dependencies, such that the agents can independently construct plans and schedules for their local sets of tasks. Because some autonomous agents (i.e., self-interested agents) might not want their constructed plans and schedules to be altered, they need to coordinate before constructing their plans and schedules. We call such a form of coordination a-priori coordination.

In the literature, two a-priori coordination approaches have been studied: one for planning, called plan coordination (Valk, 2005), and one for scheduling, called temporal decoupling (Hunsberger, 2002). In both these coordination approaches constraints are added such that the inter-agent dependencies are guaranteed to be satisfied. While this involves adding ordering constraints for plan coordination, the temporal-decoupling problem is solved by tightening the time windows in which tasks must be scheduled.

**Example 1.10.** Consider the coordination problem of Example 1.9. Those tasks can be completed by executing tasks concurrently (i.e., using multiple resources) or sequentially as shown in Figures 1.6(a) and 1.6(b).

A temporal-decoupling solution to this problem could be to constrain the refuelling tasks to be executed between 14:10 and 14:20, and adapt the time windows of the other tasks accordingly. Such a tightening of the time windows of the refuelling tasks might lead to the schedule as depicted in Figure 1.6(a). This decoupling solution allows the schedule with the shortest execution, but makes the longest execution of Figure 1.6(b) infeasible (i.e., it would violate the constraints imposed by the temporal-decoupling solution).

Another temporal-decoupling solution is to constrain the deboarding tasks to the time window \([14:00, 14:20]\), the refuelling tasks to \([14:20, 14:40]\), and the boarding
tasks to [14:40, 15:00]. This allows the agents to execute their tasks either in parallel or in sequence. When both agents prefer executing their tasks in sequence, the schedule depicted in Figure 1.6(b) results. When the agents prefer executing their tasks in parallel, idle time is introduced into the system (i.e., time in which none of the tasks is being executed).

Remark 1.11. Note that agents can prefer executing their tasks either in parallel or in sequence. A preference for either one is a strategic decision. This decision determines the ordering constraints that are added between tasks during planning. In Example 1.10, we showed that a temporal-decoupling solution may rule out a possible execution order of the tasks, or introduce system idle time.

In this example, we used temporal decoupling for achieving coordination among the agents. In fact, plan coordination could not be used, because it cannot deal with problem instances featuring time windows (i.e., it can coordinate task networks with ordering constraints only).

Temporal decoupling is dispatching preserving in the sense that all local dispatchings can be merged together to form a joint dispatching (i.e., a dispatching for the complete problem). Temporal decoupling can be applied in the dispatching phase to coordinate the agents such that they can independently construct their local dispatchings for completing their sets of tasks.

When no time windows are given (e.g., problems with qualitative-temporal constraints only), constructing a temporal decoupling for it would require arbitrary choices to be made for restricting the tasks to time windows. The local dispatchings allowed that result may not comply with the agents’ planning or scheduling preferences (i.e., preferences based on the uncoordinated situation, and not considering potential conflicts). Considering self-interested agents, the coordination mechanism
cannot be adapted to take the agents’ preferences into account, because these agents are unwilling to share their local preferences. These agents, however, do wish to add planning and scheduling constraints (reflecting their plan and schedule preferences) independently from the other agents, and they do not want these specified preference constraints to be altered. Therefore, an a-priori coordination mechanism is needed that is preference preserving in the sense that it allows the agents to specify their preference constraints and guarantees that there exists a dispatching that satisfies all preference constraints (i.e., no preferences need to be altered). Our goal is to provide such a preference-preserving coordination mechanism.

In order to achieve this goal, we have two separate phases for adding plan and schedule preferences. The agents are allowed to independently add qualitative-temporal constraints (i.e., reflecting their plan preferences) and quantitative-temporal constraints (i.e., reflecting their schedule preferences) in the planning and scheduling phase, respectively. In each phase, a coordination problem arises that needs to be solved prior to letting the agents add their local constraints. We will solve these coordination problems for the planning and scheduling phase by applying plan decoupling and schedule decoupling, respectively, as shown in Figure 1.7. By applying plan and schedule decoupling previous to autonomous planning and scheduling, we guarantee that there exists a dispatching that also satisfies all constraints added by the agents (i.e., if all agents construct local plans and schedules for which a local dispatching exists).

Example 1.12. Reconsider the coordination problem of Example 1.9. An plan-decoupling solution to this problem could be to add ordering constraints such that all deboarding tasks need to be planned before the boarding tasks. Because deboarding and boarding of the same flight are already constrained (i.e., \( D(NW142) \) before \( B(NW142) \) and \( D(NW231) \) before \( B(NW231) \)), this requires two additional ordering constraints: \( D(NW231) \) before \( B(NW142) \) and \( D(NW142) \) before \( B(NW231) \). Note, that both the shortest and longest execution of Figure 1.6 are still possible.

Our approach generalises both plan coordination and temporal decoupling. On the one hand, we extend the applicability of plan coordination to frameworks that allow richer types of constraints to be used, besides ordering constraints. On the other hand, temporal decoupling is generalised such that the agents are allowed to add a less restrictive type of scheduling constraints.
1.3 Contributions and Thesis Outline

Autonomous agents need to be able to coordinate among themselves at every level of autonomy, and at every level of decision making. Most importantly, coordination mechanisms are needed for self-interested (i.e., fully-autonomous) planning and scheduling agents. As we will show, this topic is insufficiently addressed in the available literature. The remainder of this thesis is organised as follows.

- In Chapter 2, we discuss the relevant literature on plan and schedule coordination for multi-agent systems. It turns out that the a-priori coordination-by-design approach, which we call decoupling, is the most suitable approach for coordinating self-interested agents. Finally, we point out some problems that are not studied in the available literature on decoupling.

- In Chapter 3, we define a framework for specifying sets of qualitatively-constrained tasks that are assigned to agents. This framework allows ordering and synchronisation constraints among the tasks to be specified. In this framework, we define the plan-decoupling problem. Because this problem turns out to be intractable in general, we identify two factors that influence this complexity, and determine the complexity of some subclasses by bounding either one. Finally, we provide a heuristic polynomial-time algorithm for constructing a plan-decoupling solution. This chapter is based on the following published material.


- In Chapter 4, we define a framework for specifying sets of quantitatively-constrained tasks that are assigned to agents. This framework subsumes the qualitative framework of Chapter 3, and allows time windows on tasks, temporal distances between tasks, and intervals on the task durations to be specified. We extend the definition of the plan-decoupling problem to this framework and study its complexity. Thereafter, we define the schedule-decoupling problem,
and show its strong relation with the temporal-decoupling problem (Hunsberger, 2002). This chapter is based on the following published material.


J.R. Steenhuisen and C. Witteveen. Plan coordination for durative tasks. In M.A. Salido, A. Garrido, and R. Barták, editors, Proceedings of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS), pages 73-80, September 2007. (This paper was also presented at the 5th European Workshop on Multi-Agent Systems (EUMAS), December 2007)

• In Chapter 5, we make a shift from our theoretical approach to studying the practical consequences of using decoupling. In contrast with the previous two chapters, we do not study the size of the coordination sets themselves nor the construction time required by a plan-decoupling mechanism. Instead, we focus on the effect of using plan decoupling on the plan-execution costs and plan-construction times for the agents, both for all agents together (i.e., system perspective) and for an individual agent (i.e., agent perspective). The rationale for this shift is that self-interested agents need an estimate on the costs involved with participating in a plan-decoupling mechanism. We conduct experiments in two problems for which we consider both sequential and parallel planning variants (i.e., on qualitative and quantitative task networks). This chapter is based on the following published material.


• In Chapter 6, we conclude by giving a brief summary of the contributions of this thesis, discuss the implications for situations like the example scenario’s (i.e., ground handling at Schiphol, and container transshipment in the Port of Rotterdam), and provide directions for future work.
Chapter 2

Coordinating Autonomous Planners and Schedulers

In the previous chapter, we looked at two scenario’s in which self-interested agents needed to construct plans and schedules for their local task networks (i.e., their assigned sets of tasks). We showed that, due to task dependencies, inter-agent conflicts can arise when these agents construct their plans or schedules independently from each other. We also showed two ways of coordinating the agents: By tightening the time windows within which tasks must be executed, and by adding ordering constraints among the tasks.

The goal of this chapter is twofold. First, we discuss the necessary context and background on plan and schedule coordination of a set of agents at different levels of autonomy. Second, we identify a gap in the available literature: Available coordination mechanisms for self-interested planning and scheduling agents are either defined in a too restricted framework or result in too restrictive solutions.

Although plan and schedule construction are related, they focus on different aspects. While planning is concerned with selecting and ordering a set of actions that achieve a constrained set of tasks, scheduling concentrates on time and resource allocation for completing the plan (Ghallab et al., 2004; Horling et al., 2006; Smith et al., 2000; Srivastava et al., 2001). A schedule is a refinement of a plan that meets all constraints and results in the completion of the tasks when executed, such that the goal is achieved.

In this thesis, we are interested in the coordination problems that arise when autonomous agents need to construct a joint plan or a joint schedule for achieving a joint goal. Although there exists a vast body of literature on planning and scheduling techniques, both are beyond the scope of this thesis and will therefore not be discussed here. We allow the agents to use the planning or scheduling technology of their preference. The coordination problems are studied on the abstract plans and schedules (e.g., the partial ordering of tasks) that result from the constructed concrete plans and schedules.

The remainder of this chapter is structured as follows. In Section 2.1, we provide the necessary background information on multi-agent task-based planning and
CHAPTER 2. COORDINATED PLANNING AND SCHEDULING

scheduling, and illustrate the coordination problems that arise. In Section 2.2, we provide an overview of the approaches available in the literature for coordinating autonomous planners and schedulers. In Section 2.3, we describe the available literature for coordinating self-interested agents in detail. In Section 2.4, we discuss how the available coordination approaches can be used to enable autonomous agents to specify their preferred planning and scheduling constraints. Finally, we identify a gap in the relevant literature on coordinating self-interested planning and scheduling agents, and lay out a road map for partially bridging that gap.

2.1 Multi-Agent Task-Based Planning and Scheduling

An agent can offer a limited number of services (i.e., its set of capabilities), which enables it to complete certain tasks (by executing the required actions). When achieving a goal requires a set of capabilities that is not available to any single agent, the joint effort of multiple autonomous agents together is needed. In multi-agent systems, multiple autonomous agents have decided to collectively pursue a joint goal at a certain level of autonomy (Buzing et al., 2006; Cao et al., 1997; Wooldridge, 2002).

In order to achieve their joint goal, the goal first needs to be decomposed (Castillo et al., 2005; Erol et al., 1994) into a constrained set of tasks (see box on this page on goal decomposition) which tasks are assigned to the agents such that each agent has the required capabilities for completing its local task network (i.e., its assigned set of tasks). After such an assignment is obtained, a joint plan and joint schedule need to be constructed for completing the task network, such that the agents achieve their joint goal.

**Goal decomposition** One approach to solving a task-based planning problem is by decomposition. One of the best known frameworks for achieving this is HTN planning where a decomposition relation among tasks is specified by means of a hierarchical task network (HTN) (Clement et al., 2007; Erol et al., 1994). Here, the basic problem is to construct a plan, consisting of a constrained set of primitive tasks, that achieves the goal (Ghallab et al., 2004).

An HTN is specified by a set of methods and a set of actions, called operators. The methods define the decomposition relation of the goal, or task, into a constrained set of tasks, where each task can either be primitive or non-primitive. Each non-primitive task can be decomposed further, while primitive tasks can be completed directly by actions.

An extension to HTNs is the TÆMS framework (Boddy et al., 2008; Horling et al., 2006, 1999), which allows a rich set of constraints to be used, the capabilities to be specified which are required for completing a primitive task, and a probability density function to be defined on the duration of each task.
One way of constructing such a joint plan or schedule is by letting a single party construct these, which we call the *centralised* approach. Although such a centralised approach might be acceptable for some agents, it is not when dealing with autonomous agents in general. Such situations arise in military applications, when legislation prohibits all required information to be available in one place, and when dealing with self-interested agents. Therefore, a *decentralised* approach needs to be available that enables local plans and schedules to be constructed independently.

In a multi-agent system, each agent is responsible for constructing a *local plan* and *local schedule* for completing its local task network. Additionally, all agents together are responsible for constructing a joint plan and joint schedule for achieving the joint goal. Such joint plan and schedule need to be *globally consistent* (i.e., that the task network can be completed such that all planning and scheduling constraints are satisfied). Therefore, letting each agent independently construct its local plan and local schedule sets some requirements. Clearly, the local plans and schedules at least need to be *locally consistent*, which means that each local task network can be completed according to the constructed plan/schedule such that all constraints among the tasks are satisfied. Otherwise the joint plan or schedule is guaranteed to be inconsistent, which means that it does not achieve the joint goal.

**Example 2.1.** Continuing with aircraft ground handling, consider two planes for which only passenger deboarding and refuelling need to be carried out. This situation can be specified as an HTN as shown in Figure 2.1(a). The goal can be decomposed into two non-primitive tasks for servicing each plane ($P_1, P_2$). Both these non-primitive tasks $P_i$ can be decomposed into a deboarding ($D_i$) and a refuelling ($R_i$) task. From Example 1.3, we know that an ordering constraint exists between these primitive tasks ($D$ before $R$) which is represented by an arrow between the deboarding and refuelling tasks for servicing one plane.

In this example, a deboarding and a refuelling agent are needed for completing the task network. Here, the task assignment is straightforward (i.e., each task can be assigned to only one agent), and leads to the situation depicted in Figure 2.1(b).

In this thesis, we assume a decomposition of a joint goal to be available, together with an assignment of the task network to the agents. In the remainder of this section, we will describe multi-agent task-based planning and scheduling. Because the resulting plans and schedules of all agents together also need to be consistent, we will describe the possible conflicts that can arise. How to solve these conflicts is the topic of Section 2.2.

### 2.1.1 Multi-Agent Task-Based Planning

In this section, we deal with the multi-agent task-based planning problem, which is to construct a joint plan for completing a given task network together with a task assignment to the agents. Here, each agent is responsible for constructing a local plan for its local task network. Not only must these local plans be locally consistent,
all these local plans of the different agents together must also be globally consistent such that the task network can be completed by executing this joint plan.

Each agent needs to construct a local plan for its assigned local task network, called autonomous planning, and is required to solve a local planning problem. Thereafter, these constructed local plans together need to be transformed into a joint plan through a process of plan merging (de Weerdt, 2003; Ephrati and Rosen-schein, 1993; Foulser et al., 1992; Georgeff, 1983). In Figure 2.2, a sketch is given of multi-agent planning.
2.1. MULTI-AGENT TASK-BASED PLANNING AND SCHEDULING

Intuitively, a plan forms a specification of how an agent intends to complete its assigned set of tasks. While planning is concerned with selecting and ordering of actions for completing a (local) task network, task-based planning focuses on execution ordering at the task level. Therefore, in task-based planning, a plan represents the agent’s preference on the execution order of the tasks.

Qualitative-temporal formalisms

In order to restrict the execution order of tasks, qualitative-temporal constraints can be used. Seven (or thirteen when also counting their inverse) basic qualitative-temporal constraints can be formulated between any pair of tasks (Allen, 1983): before, overlaps, during, meets, starts, finishes, and equals.

One formalism for qualitatively constraining tasks is the interval algebra (IA) (Allen, 1983). In this framework, every pair of tasks is constrained by a disjunction of the basic constraints (e.g., \( \tau_1 \{ \text{before, after} \} \tau_2 \)).

Another formalism for specifying qualitative-temporal constraints is the point algebra (PA) (Vilain and Kautz, 1986). Here, the constraints are specified between time-point variables (i.e., instead of tasks), and are a disjunction of three basic constraints: <, =, and >. In PA, each pair of time-point variables is constrained by a disjunction of these basic constraints (e.g., \( t_1 \{ <, > \} t_2 \)).

For any given task network, an important question is: What qualitative constraints can still be added while maintaining consistency? To answer this question, we need to determine the minimal network, in which all basic constraints that cannot be satisfied are removed from the disjunctions between the tasks or time-point variables. It turns out that deciding consistency in the IA is an NP-complete problem (Allen, 1983). In the PA formalism, the minimal network can be determined in polynomial time (Vilain and Kautz, 1986).

It is well-known that the basic qualitative-temporal constraints on tasks can be expressed in the PA, as shown in Figure 2.3. Due to the intractability of IA and the tractability of the PA, the focus in this thesis is on the latter.

A plan can be represented as a set of constraints among the tasks in the task network (Ghallab et al., 2004), similar to the way the constraints in the task network itself are defined (see box on this page on qualitative-temporal formalisms). More formally, a task network can be represented as a partially-ordered set \( \langle T, \rho \rangle \), where \( T \) is a set of tasks and \( \rho \) is a binary relation on \( T \), for which the following properties hold:

- (reflexivity) \( \forall a \in T : a \rho a \),
- (antisymmetry) \( \forall a, b \in T : a \rho b \land b \rho a \rightarrow a = b \), and
- (transitivity) \( \forall a, b, c \in T : a \rho b \land b \rho c \rightarrow a \rho c \).

When a plan is consistent, its set of constraints can be added to the set of constraints of the task network such that this result can again be represented as a partially-ordered set \( \langle T, \rho' \rangle \) (Valk, 2005). Here, the partial-order relation \( \rho' \) resulting after planning extends the relation \( \rho \) in the task network (i.e., \( \rho' \supseteq \rho \)).
(a) \( \tau_1 \) before \( \tau_2 \) (\( \tau_2 \) after \( \tau_1 \)).

(b) \( \tau_1 \) overlaps \( \tau_2 \)

(\( \tau_2 \) overlapped by \( \tau_1 \)).

(c) \( \tau_1 \) during \( \tau_2 \) (\( \tau_2 \) contains \( \tau_1 \)).

(d) \( \tau_1 \) meets \( \tau_2 \) (\( \tau_2 \) met by \( \tau_1 \)).

(e) \( \tau_1 \) starts \( \tau_2 \) (\( \tau_2 \) started by \( \tau_1 \)).

(f) \( \tau_1 \) finishes \( \tau_2 \)

(\( \tau_2 \) finished by \( \tau_1 \)).

(g) \( \tau_1 \) equals \( \tau_2 \).

Figure 2.3: Representation of the basic qualitative-temporal constraints.
Such a partially-ordered set $\langle T, \rho \rangle$ can be represented as a directed acyclic graph (DAG) $\langle V, A \rangle$. Basically, all elements $a, b \in T$ have a corresponding vertex $v_a, v_b \in V$, and every relation $a \rho b$ is represented by a directed arc $v_a \rightarrow v_b$. Representing all binary relations in $\rho$, however, would cause an abundance of arcs to be drawn in the DAG representation. Therefore, we only draw the arcs associated with a transitive reduction$^1$ $\rho^-$ of the partial-order relation $\rho$.

After the agents have constructed their local plans, these plans need to be merged into a joint plan for the original task network. Simply merging these local plans without any changes to the constructed plans does not necessarily result in a joint plan that is consistent. In fact, even when each constructed local plan is locally consistent, some constraints in the task network that were not considered by any of the agents might be violated by the joint plan.

![Figure 2.4: A plan-uncoordinated task network.](image)

**Example 2.2.** Consider the simple example depicted in Figure 2.4. In Figure 2.4(a), two agents $A_1, A_2$ are assigned two tasks each, $\tau_1, \tau_4$ and $\tau_2, \tau_3$, represented by the vertices. Furthermore, there are two ordering constraints $\tau_1$ before $\tau_2$ and $\tau_3$ before $\tau_4$, represented by arcs.

Here, an inconsistent joint plan can result even though both agents construct locally-consistent plans, as is illustrated in Figure 2.4(b). This situation occurs when agent $A_1$ constructs a plan that can be represented as $\tau_4$ before $\tau_1$, and agent $A_2$ constructs the plan $\tau_2$ before $\tau_3$. Clearly, both plans are locally consistent, but are jointly inconsistent because this would require $\tau_1$ to precede $\tau_2$, but also vice versa.

As illustrated by Example 2.2, the joint plan can become inconsistent due to an interaction of the local plans through the inter-agent constraints that cannot be satisfied (i.e., a cyclic ordering). Although such an inconsistent joint plan is not guaranteed to result from independent planning, we call a task network plan uncoordinated when there exists a combination of local plans that together result

---

$^1$Given a binary relation $\rho$ on a set $\chi$. The transitive closure $\rho^+$ of $\rho$ is the transitive relation on set $\chi$ that contains $\rho$ and is minimal. The transitive reduction $\rho^-$ of $\rho$ is a minimal relation on $\chi$ such that the transitive closure of $\rho^-$ is the same as the transitive closure of $\rho$. 

in an inconsistent joint plan. For constructing a consistent joint plan, some form of plan coordination is needed that manages the inter-agent dependencies: guarantees the satisfiability of the joint plan.

In this section, we have looked at multi-agent planning. Specifically, we looked at multi-agent systems where the agents employ autonomous planning. We saw that decentralised plan construction, in which each agent independently constructs a locally-consistent plan for its local task network, might result in conflicts among these plans. The question we need to answer is how such independently constructed plans can be merged into a globally-consistent joint plan.

For now, assume that this problem has been solved and the agents successfully constructed a consistent joint plan. In Section 2.2, we will look at how the plan-coordination problem has been addressed in the literature.

![Diagram of multi-agent task-based scheduling](image)

**Figure 2.5:** Multi-agent task-based scheduling.

### 2.1.2 Multi-Agent Task-Based Scheduling

In the previous section, we discussed multi-agent task-based planning and the problem of plan coordination. After the agents succeeded in constructing a consistent joint plan, a scheduling problem needs to be solved before starting with the dispatching phase. In this section, we deal with this multi-agent task-based scheduling problem, which is to construct a joint schedule for completing the task network according to the constructed joint plan.
2.1. MULTI-AGENT TASK-BASED PLANNING AND SCHEDULING

Again, each agent is responsible for constructing a local schedule for its local task network, that also needs to satisfy the constructed local plan. After this autonomous scheduling by the agents, the joint schedule can be constructed by merging all local schedules (Tsamardinos et al., 2000). An overview of this multi-agent task-based scheduling scenario is given in Figure 2.5.

Intuitively, a schedule specifies when an agent intends to execute its assigned set of tasks (i.e., when execution may start and when it should be completed). Such a schedule represents the agent’s preference on the execution time and is a refinement of its plan, in the sense that all constraints in the plan should be satisfied.

Quantitative-temporal reasoning Quantitative-temporal constraints allow the temporal distance between events (i.e., time-point variables) to be restricted. In the general temporal constraint networks, used in the formulation of a temporal constraint satisfaction problem (TCSP) (Dechter et al., 1991), the temporal distance between any pair of time-point variables can be constrained by a union of allowed intervals. For instance, two time-point variables $t_1, t_2$ can be constrained such that $t_2 - t_1 \in [1,4] \cup [6,7]$ must hold (i.e., $t_2$ takes place after $t_1$ with a certain distance). Similar to the situation in IA, determining the minimal network for task networks that use these types of constraints requires solving an NP-complete problem (Dechter et al., 1991).

Contrary to the representation of plans, schedules cannot be represented by qualitative constraints, because the scheduling constraints involve absolute moments in time. Clearly, the qualitative-temporal formalisms for planning do not suffice for scheduling, and a quantitative-temporal formalism is needed for representing a task network (see box on this page on quantitative-temporal reasoning).

Due to the intractability of TCSP, a tractable subclass is often used instead: Simple Temporal Networks (STN) (Dechter et al., 1991).

Definition 2.3 (Simple Temporal Networks). A Simple Temporal Network $\mathcal{S}$ is a tuple $(T, \mathcal{C})$, where $T$ is a set of time-point variables $\{t_1, t_2, \ldots, t_n\}$, and $\mathcal{C}$ is a finite set of binary constraints $\{c_1, c_2, \ldots, c_m\}$ on the variables in $T$ with each constraint $c_i \in \mathcal{C}$ having the form $t - t' \leq \delta_i$, for some number $\delta_i \in \mathbb{R}$.

In such STNs, every pair of time-point variables $t_i, t_j$ is constrained by a single interval $-\delta_{ji} \leq t_j - t_i \leq \delta_{ij}$ (i.e., a lower bound $t_i - t_j \leq \delta_{ji}$ and an upper bound $t_j - t_i \leq \delta_{ij}$ on the temporal distance). Note that all these constraints are relative to other time-point variables. Adding absolute constraints, called time windows, for constraining the values is commonly done by introducing a fixed-reference point $z$ which is fixed to a certain value (e.g., $z = 0$), such that each absolute constraint $t \in [a, b]$ can be translated to relative constraint $t - z \in [a, b]$. Instead of representing an STN as a set of inequalities, it can be represented as a distance-interval graph (see Figure 2.6(a)) or as an distance graph (see Figure 2.6(b)). Although both representations are equivalent, as shown in the next example, the compact distance-interval-graph representation will be used in this thesis.
Example 2.4. Consider an STN with two time-point variables \(t_1, t_2\), which are constrained by \(t_2 - t_1 \in [1, 4]\), and absolute time windows \(t_1 \in [1, 5]\) and \(t_2 \in [2, 9]\). This STN can straightforwardly be represented as the distance-interval graph shown in Figure 2.6(a). In the distance graph, all arcs are labelled with a single value, which is the upper bound on the temporal distance. For instance, the constraint \(t_2 - t_1 \in [1, 4]\) between the two time-point variables becomes \(t_1 - t_2 \leq -1\) and \(t_2 - t_1 \leq 4\) which result is shown in Figure 2.6(b). To represent the absolute time windows on \(t_1, t_2\), we introduce the fixed-reference point \(z = 0\) and translate the time-window constraints accordingly.

Let us move on from schedule representation to schedule construction. Given a consistent joint plan, a schedule needs to be constructed for executing the tasks somewhere in time and according to the plan. Efficient schedules are likely to be preferred that, for instance, minimise the shortest time from start to finish, called *makespan*.

In the literature, two types of scheduling can be found: *dispatch scheduling* (or dispatching) and *partial-order scheduling*. In dispatch scheduling (Garey and Johnson, 1979), the constructed schedule specifies fixed starting times for each task. In partial-order scheduling (Policella et al., 2007), the schedule does not necessarily fix the starting times for each task, but specifies a *time window* in which a task needs to be started. By only fixing the starting times of the tasks on the critical path, such a schedule is more *flexible* without increasing the makespan.

After the agents have constructed their local schedules, the joint schedule for executing the joint plan for completing the task network can be constructed by merging the local schedules. Obviously, the local schedules constructed by the agents need to be locally consistent. However, similar to multi-agent planning, the joint schedule may become inconsistent due to an interaction of the local schedules with the inter-agent constraints.

To illustrate this, consider the simple example depicted in Figure 2.7(a). Here, two agents \(A_1, A_2\) are assigned two time-point variables, \(t_1\) and \(t_2\), where time point \(t_1\) needs to be executed in the time interval \([1, 5]\), time point \(t_2\) in \([2, 9]\), and time point \(t_2\) needs to be executed at least 1 and at most 4 time units after \(t_1\) (i.e., \(t_2 - t_1 \in [1, 4]\)).
2.2 Coordinating Agents’ Plans and Schedules

In the previous section, we looked at multi-agent planning and scheduling where each agent independently constructs a plan or schedule for its assigned set of tasks. While the constructed plans and schedules are locally consistent, the agents’ plans and schedules together also need to be consistent (i.e., globally consistent). The question that remains to be answered is

“How can independently constructed plans and schedules be merged into a globally-consistent joint plan and schedule, respectively?”

We saw that the basic answer to this question is that a coordination problem needs to be solved, that manages the inter-agent constraints (Malone and Crowston, 1994). For solving such coordination problems, a coordination mechanism is needed (Christodoulou et al., 2004).

While each coordination mechanism solves some coordination problem, the autonomous nature of the agents might label a mechanism as unacceptable (Jennings, 1996; Wooldridge and Jennings, 1995). In fact, this acceptability depends on the agent’s level of autonomy (Barber and Martin, 2002) which can be either obedient, cooperative, or self interested (Sellner et al., 2006; van der Vecht et al., 2006). Obedient agents accept all changes made by the mechanism to its constructed plans or schedules without considering alternatives. Cooperative agents truthfully share their decisions with and trust all information from the mechanism, and are willing to adapt their individual decisions, but not necessarily accepting the changes proposed.
by the mechanism. Finally, agents can be self interested in the sense that they are unwilling to revise their independently-constructed plans and schedules.

Hence, self-interested agents need to be coordinated before they construct their local plans and schedules, while cooperative agents also allow coordination during planning and scheduling (i.e., cooperative problem solving). Obedient agents even accept that their constructed plans and schedules are coordinated afterwards by replacing it by a new solution (e.g., constructed with the centralised approach). Therefore, we classify the coordination mechanisms based on the moment at which the coordination problem is solved.

In the following three sections, we look at the available coordination mechanisms for solving the coordination problems a posteriori, a interiori, and a priori with respect to the independent planning or scheduling by the agents.

2.2.1 A-Posteriori Coordination

One coordination approach is to let the agents construct local plans and schedules and then apply a coordination mechanism that resolves all conflicts (i.e., when there are any conflicts at all) by revising the constructed plans or schedules. Such a mechanism can additionally be used to increase efficiency by exploiting positive interactions between the anticipated actions of the agents. From a computational point of view, this a-posteriori approach is preferred when conflicts are (very) unlikely to occur and coordination verification is relatively cheap. This approach is called merging, and has been studied widely (de Weerd, 2003; Ephrati et al., 1995; Georgeff, 1983; Tsamardinos et al., 2000; von Martial, 1992).

In plan merging, the combination of the independently-constructed (partial) plans is considered. One approach then is to resolve conflicts and other negative interactions by adapting the joint plan, or even improving the joint plan by exploiting positive interactions (von Martial, 1992). In a related approach (Foulser et al., 1992; Yang et al., 1992), restrictions are added to the local plans such that efficient merging is guaranteed. Improving on this work, an iterative approach to repairing an inconsistent multi-agent partial-order plan was proposed that significantly reduces the computation time (Cox and Durfee, 2005). These approaches require that the agents communicate (abstractions of) their constructed plans to some central authority, and comply to the changes it proposes.

In schedule merging, the combination of independently-constructed schedules is considered. To identify conflicts, an algorithm has been developed (Tsamardinos et al., 2000) that merges these independently-constructed schedules into a conditional STNs (i.e., an extension of simple temporal networks). This algorithm gives a set of constraints (i.e., on the conditions in the conditional STN) that needs to be satisfied in the joint schedule.

These merging approaches suffer at least from the following two disadvantages. First, the agents need to (be willing to) share their plans/schedules with the coordination mechanism. Second, the agents must (be willing to) revise their constructed solutions according to the proposition of the coordination mechanism. These re-
requirements might be unacceptable for the participating agents. For instance, cooperative agents might not be willing to revise their plans or schedules according to the mechanism’s proposition.

### 2.2.2 Interleaved Coordination

Instead of separating the local planning and scheduling from the coordination thereof, we could interleave these processes.

In plan-combination search (Ephrati and Rosenschein, 1994), agents iteratively propose the changes to the world they can make with a single action from their partial plans (i.e., the set of all consistent plans that reach the goals). These sets of proposed agent actions are ranked using a heuristic from which the best candidate set is chosen. A variant of this iterative plan-merging approach, where no central authority is used, is to let the agents vote on their joint actions (Ephrati et al., 1995).

In HTNs, independent planning and plan merging can be done at different levels in the hierarchy, and extending the plans iteratively (Corkill, 1979). In this approach, conflicts are resolved by taking choices on ordering and synchronisation at abstract levels, which unfortunately can lead to unresolvable conflicts at a more detailed level such that backtracking is needed. In order to circumvent this backtracking problem, possible plan interferences must be detected before fully extending these plans. This can be achieved by having the agents communicate plan summaries from the hierarchical plan operators, when using HTN planners (Clement and Durfee, 1999).

Instead of only resolving conflicts, positive interactions between agents can also be exploited (i.e., improving plan quality). Using HTNs, some work has been done in this direction by determining how agents can help each other achieving their goals (von Martial, 1992). Here, the agents exchange their individually-constructed plans at increasing levels of detail. In this way, agents can help each other while pursuing their own goals, thereby reducing plan inefficiencies. By letting the agents communicate with others directly, each agent can construct a partial global plan (i.e., an agent’s partial plan merged with that of others). Based on these partial global plans, the agents can iteratively adapt their local plan and communicate it again (des Jardins et al., 2000; Durfee and Lesser, 1991; Lesser et al., 2004).

For schedule coordination of tasks with precedence constraints, a coordination mechanism has been presented that is capable of producing high-quality solutions for real-time job-shop scheduling (Liu and Sycara, 2003). For coordinating teams of agents with tasks that also require coordination in the dispatching phase (i.e., during execution), a market-based approach was used for which good performance was reported (Kalra et al., 2006, 2004).

These interleaved coordination approaches suffer from similar disadvantages as the merging approaches (i.e., they require sharing and revising their local plans or schedules). For the interleaved approaches, communication costs are even higher because the plans and schedules are communicated multiple times (Clement and
These disadvantages might be unacceptable for the participating agents. In a competitive environment, for example, self-interested agents are unlikely to be willing to share their constructed plans with others, let alone to revise them.

### 2.2.3 A-Priori Coordination

In the previous coordination approaches, the merging step was used for detecting conflicts which were then solved by changing the constructed plans. Although these approaches are guaranteed to result in consistent joint plans and schedules, they require changes to be made to the local plans and schedules. Taking an a-priori coordination approach, on the other hand, solves the coordination problem before the agents individually construct their local plans or schedules. Instead of resolving problems, all potential conflicts are prevented such that merging does not require any revisions to be made.

This a-priori coordination approach can be further divided into three categories.

1. **Social laws**: Construct rules to prevent all possible conflicts by specifying the allowed actions for all possible situations.

2. **Task assignment**: Assign the tasks to the agents such that the dependencies among the agents guarantee that no conflicts can arise.

3. **Coordination by design**: Add constraints to a given task assignment such that no conflicts can arise among the agents in the adapted problem.

In the following sections, we will describe these three categories in more detail.

### Social Laws

Probably the most commonly used a-priori coordination approach in practice is that of social laws (Shoham and Tennenholtz, 1995). In the traffic domain, for example, one such a collection of social laws is the set of traffic rules, in which it is, among other things, regulated on which side of the road to drive. In general, when such social laws exist in a certain environment, they constrain the agents in that environment and eliminate all potential conflicts (i.e., assuming all agents adhere to these social laws).

Besides the application in traffic management, social laws have been successfully used in other domains. In distributed delivery (Goldman and Rosenschein, 1994), for instance, where coordinated behaviour results from helping others complete their tasks by picking up goods and bringing them closer to their destination, or to the route of the responsible transporting agent.

Social laws provide *domain-dependent solutions* to coordination problems that are applicable to multiple problems. Not only do social laws guarantee coordination before the agents start planning or scheduling, they even guarantee coordination before the agents know the exact problem. Because these social laws are domain-dependent—and not problem dependent—rules for preventing conflicts, they have
to guarantee conflict-free solutions for all problems in a certain domain. Therefore, this approach may constrain the agents too much in certain specific instances of problems.

**Task Assignment**

By constructing *problem-dependent solutions* for plan and schedule coordination (cf. domain-dependent solutions), more efficient coordination solutions can be achieved. One such problem-dependent coordination approach is by solving the *task-assignment problem* such that no further coordination problem needs to be solved (i.e., the task assignment ensures coordination). Here, the fact is used that the necessity for coordinating an instance is based on the inter-agent dependencies induced by the task assignment. When no such inter-agent dependencies exist, then no further coordination is needed.

Therefore, the basic problem is to find a task assignment to agents in which there are no inter-agent constraints. If there are no constraints among the tasks that are assigned to different (teams of cooperating) agents, we call these agents to be a set of *loosely-coupled agents*. Deciding whether there exists a task assignment that results in a set of loosely-coupled agents turns out to be NP-complete, also when the tasks are constrained with only precedence relations (Shehory and Kraus, 1996, 1998).

Although coordination can be achieved through task assignment for some instances, it requires solving a computationally hard problem. However, the existence of a coordinated task assignment is not guaranteed, and therefore, a coordination problem remains to be solved. The following classification can be made based on the degree to which further coordination is needed among the agents, based on the inter-agent constraints resulting from the task assignment (Zlot and Stentz, 2003, 2006).

- **Loosely-coupled agents**: No coordination problem remains to be solved when no agent is assigned a task with a dependency with some task assigned to another agent.

- **Moderately-coupled agents**: Coordination in the planning phase is needed when the task assignment of a partially-ordered set of tasks results in inter-agent dependencies (i.e., precedence constraints).

- **Tightly-coupled agents**: Coordination in both the planning and the scheduling phase is needed when the inter-agent dependencies resulting from the task assignment include synchronisation constraints.

**Coordination by Design**

When task assignment does not result in a set of coordinated agents (i.e., they are moderately- or tightly coupled), a coordination problem remains to be solved. Instead of coming up with a domain-dependent solution for the coordination problem, we can construct problem-dependent coordination mechanisms. An advantage
of constructing such coordination mechanisms is that it results in instance-specific solutions, comprising fewer constraints for the agents compared to solutions from domain-dependent mechanisms. A coordination-by-design (ter Mors et al., 2009) approach does exactly this: solving a coordination problem for a given instance by posing additional constraints on the agents.

For planning, plan coordination (Valk, 2005) has been studied on task networks with precedence constraints among the tasks. The defined and studied plan-coordination problems take a partitioned set of tasks with precedence constraints among them as their input. Then, a set of coordination constraints is added to the instance, such that it becomes plan coordinated. Although the decision variant of the cardinal-minimal plan-coordination problem is proven to be intractable in general, a polynomial-time algorithm is provided for constructing a sufficient coordination set.

For scheduling, temporal decoupling (Hunsberger, 2002) has been defined and studied for dispatch scheduling, using simple temporal networks (STNs). In such STNs, the distance between each pair of time-point variables is constrained by a lower and an upper bound. For solving these instances, the intervals of dispatch values are changed (i.e., tightened) for all time-point variables involved in an inter-agent constraint, such that this constraint is guaranteed to be satisfied with the tightened intervals. A polynomial-time algorithm is provided, which constructs a subset-minimal solution such that the agents are coordinated for independent dispatching.

Contrary to the other coordination approaches discussed in this section, coordination by design is applicable to sets of autonomous agents at any level of autonomy (i.e., even for self-interested agents). Moreover, coordination is guaranteed to be achieved with this approach, with a coordination set that is specific for the instance at hand. Due to the general applicability, these coordination-by-design mechanisms can be used when there is no other—more specific—coordination mechanism available. As such, these plan and schedule coordination mechanisms enable planning and scheduling agents at any level of autonomy. Therefore, we will take a closer look at plan coordination and temporal decoupling in the following section.

2.3 A-Priori Coordination of Autonomous Planners and Schedulers

While coordination problems can be solved for agents at different levels of autonomy, the most challenging coordination problems are for self-interested agents. These agents maintain the highest level of autonomy, and pose the hardest constraints on solving the coordination problems. The a-priori coordination-by-design approach is acceptable for coordinating self-interested planning and scheduling agents.

In the remainder of this section, we will describe two such a-priori coordination-by-design approaches for planning and scheduling agents in detail.
2.3.1 Plan Coordination

Plan Coordination (Valk, 2005) prevents planning conflicts in task networks, where tasks can be constrained by precedence relations. We saw that, when dealing with such task networks, consistent local plans can result in a directed inter-agent cycle (see Section 2.1.1). The approach taken here is to reduce the planning freedom of the agents such that all conflicts are prevented, by adding a set of intra-agent precedence constraints, called the coordination set. Although this restricts the individual planning freedom, it does provide the agents with the guarantee that their constructed plans do not need to be revised (i.e., assuming all agents construct plans that are not in conflict with the given constraints).

Example 2.5. Reconsider the plan-uncoordinated task network of Example 2.2 (also shown in Figure 2.8(a)). Here, an uncoordinated task network is shown with two agents $A_1, A_2$, each assigned two tasks, $\tau_1, \tau_4$ and $\tau_2, \tau_3$, respectively, that are constrained by the precedence constraints $\tau_1 \prec \tau_2$ and $\tau_3 \prec \tau_4$, which are represented by the solid arcs. In Example 2.2, we showed that one potential conflict exists, being the directed inter-agent cycle $\langle \tau_1, \tau_2, \tau_3, \tau_4, \tau_1 \rangle$. Now, this plan-coordination problem can be solved by adding the single coordination constraint $\tau_1 \prec \tau_4$ as shown in Figure 2.8(b).

When dealing with self-interested planning agents, the plan-coordination problem should be solved by reducing the planning freedom in a minimal way. Although a reduction in planning freedom due to plan coordination is unavoidable, some optimisation criterion is needed for finding the desired coordination set.

Example 2.6. Reconsider the situation of Example 2.5 with agent $A_1$ being assigned an additional task $\tau_5$ which is constrained by $\tau_4 \prec \tau_5$. Now, compare the following two equally-sized coordination sets: $\Gamma_1 = \{\tau_1 \prec \tau_4\}$ and $\Gamma_2 = \{\tau_3 \prec \tau_2\}$. When the agents are coordinated by coordination set $\Gamma_1$, agent $A_1$ has no planning freedom left because its tasks are totally ordered, and agent $A_2$ can still plan either $\tau_3 \prec \tau_2$ or $\tau_2 \prec \tau_3$ (i.e., there are 2 possible plans). When coordination set $\Gamma_2$ is used, agent $A_2$
has no planning freedom left, while agent $A_1$ can choose either $\{\tau_1 \prec \tau_4\}$, $\{\tau_5 \prec \tau_1\}$, and $\{\tau_4 \prec \tau_1, \tau_1 \prec \tau_5\}$ (i.e., there are 3 possible plans). Clearly, the coordination sets result in different reductions in planning freedom.

We are interested in the plan-coordination variant that minimises the size of the coordination set. For this minimal criterion, there are multiple measures available, the applicability of which depends on the problem considered. For instance, the plan-coordination mechanism can construct a cardinal-minimal coordination set (i.e., smallest set of additional constraints), or a coordination set which is subset minimal in the sense that leaving one constraint out results in an uncoordinated situation. While a cardinal-minimal coordination set necessarily is subset minimal, the reverse relation does not hold as illustrated by the following example.

![Figure 2.9: A plan-uncoordinated task network.](image)

**Example 2.7.** Consider the plan-uncoordinated task network depicted in Figure 2.9. Here, three agents $A_1, A_2, A_3$ are assigned two tasks each, where each agent has one task with only incoming precedence constraints and one task with only outgoing precedence constraints. Because the task network is plan uncoordinated a coordination set for it consists of at least one constraint. Now, consider the following two coordination sets: $\Gamma_1 = \{\tau_1 \prec \tau_6, \tau_4 \prec \tau_3\}$ and $\Gamma_2 = \{\tau_5 \prec \tau_2\}$. It can easily be shown that both coordination sets are subset minimal, because leaving either constraint from $\Gamma_1$ allows a directed inter-agent cycle (and coordination set $\Gamma_2$ already has minimum cardinality).

The computational complexity of the plan-coordination problem with minimum cardinality has been studied (Valk, 2005). It turns out that deciding whether an instance is plan decoupled already is coNP-complete (Valk, 2005). When the original task network is consistent (i.e., there are no directed inter-agent cycles), then the existence of a coordination set consisting of precedence constraints is guaranteed. The decision variant of the cardinal-minimal plan-coordination problem is $\Sigma_p^2$-complete (Valk, 2005).

Because this cardinal-minimal plan-coordination problem is intractable, we should look for polynomial-time approximation algorithms for solving it instead. Concerning the approximability, it was shown that this plan-coordination problem is APX-hard (Valk, 2005).
2.3. A-PRIORI COORDINATED PLANNING AND SCHEDULING

There exists a simple polynomial-time heuristic that succeeds in finding a sufficient—but not necessarily minimum—coordination set (Valk, 2005). This algorithm uses the fact that the tasks are partially ordered by a precedence relation, which is necessarily acyclic. Therefore, there exists a task that is pre-requisite free (i.e., does not have any tasks that must be completed previously) such that it can be executed first. After completing this task, other tasks might become pre-requisite free such that we can continue iteratively until all tasks have been completed.

The performance was empirically studied in terms of the plan cost and the plan-construction time. Using a planning problem called LOGISTICS (Bacchus, 2001), the performance achieved on solving the instances after plan coordination was compared with solving them centralised (i.e., without plan coordination). It turned out that the plan-coordinated planners clearly outperformed all centralised planners used (Valk, 2005).

**Remark 2.8.** Note that, during both plan coordination and local planning, no schedules nor dispatchings are considered for the tasks. Instead, the agents are interested in the ordering of the tasks, with the only restriction that the task network can still be completed (i.e., remains consistent) with these additional ordering constraints.

In the literature, plan coordination is studied for instances where the agents have been assigned a set of tasks with precedence constraints among them. Therefore, it provides an a-priori coordination solution only for loosely and moderately-coupled agents. However, agents can easily become tightly coupled, when they are assigned a task network which contains other types of constraints (e.g., synchronisation constraints). Then, existing plan-coordination techniques are not sufficient for coordinating such tightly-coupled agents.

2.3.2 Temporal Decoupling

*Temporal decoupling* (Hunsberger, 2002) prevents dispatching (dispatch scheduling) conflicts such that merging any set of local dispatchings results in a joint dispatching, using simple temporal networks (STNs). We saw that a set of locally-feasible dispatchings can jointly be infeasible due to an interaction through the inter-agent constraints (see Section 2.1.2). The approach taken here is to reduce the dispatching flexibility of the agents such that all dispatching conflicts are prevented. Although this restricts the dispatching flexibility, it does provide the agents with the guarantee that their constructed dispatchings for their sub-STNs do not need to be revised.

The temporal-decoupling problem considers each inter-agent constraint and adapts the time windows of the involved time-point variables in such a way that the constraint under consideration is guaranteed to be satisfied. The margins by which the time windows need to be tightened can be determined easily by considering the lower and upper bound of the inter-agent constraints separately. Now, the time windows of both time-point variables must be adapted in such a way that their tightenings together should at least equal the determined margin. The next example might help in understanding this approach.
Example 2.9. In Figure 2.10(a), an uncoordinated STN is depicted. There exist two potential conflicts, in the sense that dispatchings can violate either the lower or the upper bound of the inter-agent constraint $t_2 - t_1 \in [1, 4]$. Consider the upper bound $t_2 - t_1 \leq 4$, which must be guaranteed to be satisfied even when the earliest dispatching for $t_1 \in [1, 5]$ and latest for $t_2 \in [2, 9]$ are chosen. While this is not guaranteed initially (i.e., because $9 - 1 \not\leq 4$), we can calculate the margins $m_1, m_2$ by which the time windows need to be adjusted by solving $(9 - m_2) - (1 + m_1) \leq 4$, which can be rewritten into $4 \leq m_1 + m_2$. For instance, using margins $m_1 = 1, m_2 = 3$ satisfies this inequality and can result in the tightening of the time windows $t_1 \in [1 + m_1, 5] = [2, 5]$ and $t_2 \in [2, 9 - m_2] = [2, 6]$. The potential conflict with the lower bound $1 \leq t_2 - t_1$ can be solved similarly, such that the time windows are tightened as shown in Figure 2.10(b). Note that now any combination of local dispatchings chosen by agents $A_1$ and $A_2$ satisfies the inter-agent constraint.

When dealing with self-interested agents, the temporal-decoupling problem should be solved by reducing the dispatching freedom in a minimal way. A smaller coordination set does not necessarily result in a smaller reduction in dispatching freedom. Nevertheless, we are interested in the temporal-decoupling variant that minimises the size of the coordination set. However, for this minimality criterion, there are again multiple measures available, the applicability of which depends on the problem considered. For instance, a temporal-decoupling mechanism can construct a tightening which has minimum cardinality (i.e., smallest sum of constraint tightenings), or one that is subset minimal in the sense that loosening one constraint results in an uncoordinated situation.

The subset-minimal temporal-decoupling problem has been studied and polynomial-time algorithms have been described (Hunsberger, 2002). Concerning the experimental evaluation of the developed algorithms for finding solutions to the temporal-decoupling problem, different strategies were compared. In these experiments, the required computation times and the loss of flexibility (i.e., the sum of all constraint tightenings) were measured. One strategy turned out to be able to find solutions that hardly reduced the flexibility.

The cardinal-minimal temporal-decoupling problem is also polynomially solvable. This result is a consequence of the proof given by Planken et al. (2010) that the–more generalised–optimal temporal-decoupling problem (which minimises the summed loss
2.4 DISCUSSION

of flexibility, and is NP-hard in general) can be formulated as a linear-programming (LP) problem (Dantzig and Thapa, 2003) when the performance metric is linear. Because the performance metric of the cardinal-minimal temporal-decoupling problem is linear (i.e., the sum of the amount by which the constraints are tightened), the problem can be formulated as an LP problem and solved in polynomial time (Khachiyan, 1979, 1980).

Temporal decoupling assumes that every solution to the partitioned STN (i.e., a dispatching that satisfies all temporal constraints) satisfies all constraints of the scheduling problem. Specifically, it is assumed that each solution satisfies all resource constraints (i.e., the STN is assumed to be resource consistent). Often, no resource-consistent STN is available for multi-agent scheduling problems with limited resources, because the resource constraints are an important part of the scheduling problem. In those cases, before temporal decoupling can be applied, a scheduling problem needs to be solved that constructs the required resource-consistent STN.

Note that, scheduling agents can construct schedules that specify exact starting times of tasks, which is satisfied by only one dispatching. These agents require that such constructed dispatchings do not conflict with those constructed by other agents. This requirement differs from the requirement set by the planning agents, where only the consistency of the joint plan was required.

Finally, note that temporal decoupling requires time windows to be specified for every task, while such time windows are not allowed in the framework used for plan decoupling. Hence, temporal decoupling is not suitable for achieving plan decoupling.

2.4 Discussion

In this chapter, we considered sets of autonomous agents that need to construct plans and schedules for achieving a joint goal which they decided to pursue. We described how this joint goal can be decomposed into a constrained set of tasks, called a task network, such that completing all tasks—satisfying all constraints—achieves the joint goal. We assume such a decomposition to be given together with a task assignment such that each agent is capable of and responsible for completing its set of tasks.

Given such a task network together with an assignment thereof to the agents, a joint plan and joint schedule need to be constructed for completing it. Because each agent is responsible for completing its assigned local task network, it needs to construct a local plan and local schedule for it. Of course, these local plans and schedules must be locally consistent. Furthermore, all these constructed local plans and schedules (of the different agents) together must also be globally consistent. The task network can be completed as planned and scheduled if and only if both local and global consistency hold.

Because independently constructed local plans and schedules together might be globally inconsistent (i.e., even when they all are locally consistent), coordina-
CHAPTER 2. COORDINATED PLANNING AND SCHEDULING

tion problems need to be solved. In the literature, many coordination mechanisms are available that solve plan or schedule-coordination problems either a-priori, a-interiori, or a-posteriori with respect to local plan or schedule construction.

While coordination problems can be solved for agents at different levels of autonomy, the most challenging coordination problems are those for self-interested agents. Because these agents maintain the highest level of autonomy, they pose the hardest constraints on solving the coordination problems. When a coordination mechanism is acceptable for self-interested agents, then it is—from an autonomy point of view—acceptable for agents at any level of autonomy. Therefore, this thesis focuses on self-interested planning and scheduling agents.

The goal of this thesis is to enable a set of self-interested agents to specify their planning and scheduling preferences as represented in Figure 2.11. We will represent these planning and scheduling preferences as qualitative- and quantitative-temporal constraints, respectively. Self-interested agents require that they can specify such local preferences independently from the other agents, and that they do not need to alter their preferences. Hence, preference-preserving plan and schedule-coordination mechanisms are needed in the sense that they allow the agents to specify their preference constraints and guarantee the existence of a dispatching that satisfies all preference constraints (i.e., no preferences need to be altered).

Because self-interested agents do not want to revise their constructed plans or schedules, only a-priori coordination-by-design approaches are acceptable. In the previous section, two such approaches were described: plan coordination (Valk, 2005) and temporal decoupling (Hunsberger, 2002). Both approaches achieve coordination by casting a set of additional constraints, called a coordination set, onto the task network.

By adding a coordination set to the task network, agents can be coordinated for independent planning and scheduling. This means that the agents are decoupled, in the sense that the coupling among the agents with respect to their planning or scheduling constraints is eliminated.

In the remainder of this thesis, we will use plan decoupling and schedule decoupling to denote the preference-preserving coordination mechanisms for the planning and scheduling phase, respectively. The relation to the literature is best described as follows. Plan coordination (Valk, 2005) is plan decoupling when considering task networks that is constrained by precedence constraints only; We include synchronisation constraints. Temporal decoupling (Hunsberger, 2002) is schedule decoupling when the agents are allowed to specify one dispatching only (i.e., exact starting times of all tasks); We take into account that agents want to specify intervals of dispatchings.

Eventually, the agents need a joint dispatching that specifies the exact starting times of each task. For this purpose, temporal decoupling (Hunsberger, 2002) can be used to enable self-interested agents to construct local dispatchings independently.
Figure 2.11: Coordinated planning and scheduling by self-interested agents.
2.5 Research Questions and Thesis Outline

In the previous section, we presented the goal of this thesis: Enabling self-interested agents to specify their planning and scheduling preferences. We argued that only plan and schedule decoupling are suitable coordination mechanisms for this purpose, because these are—from an autonomy point of view—acceptable for agents at any level of autonomy. In Section 2.3, we described the available mechanisms for plan and schedule decoupling (i.e., plan coordination and temporal decoupling). In this section, we will identify a gap in the literature on plan and schedule decoupling, formulate research questions, and provide an outline for the remainder of this thesis to (partially) bridge this gap.

Plan and schedule decoupling have much in common, but also have many differences. First, the frameworks in which they have been studied differ: While plan decoupling has been studied for task networks without time windows (i.e., precedence constraints only), schedule decoupling requires time windows to be specified for each task. Second, plan decoupling allows the agents to impose ordering constraints on their tasks, while temporal decoupling allows the agents to construct a dispatching for their tasks (i.e., fixing the starting time of each task).

We have come to a gap in the existing literature concerning coordination mechanisms for self-interested planning and scheduling agents. On the one hand, plan decoupling cannot yet be applied when tasks are constrained by constraints other than precedence constraints (e.g., synchronisation constraints and time windows). Schedule decoupling, on the other hand, cannot yet be used for a less restrictive type of scheduling than dispatching (i.e., dispatching preserving instead of preference preserving).

For bridging this gap, we formulate the following research questions. First, we want to extend the applicability of plan decoupling to task networks with richer qualitative- and quantitative-temporal constraints, and allowing richer types of qualitative-temporal constraints to be added.

**Research question 1**

Is it possible to extend the applicability of plan decoupling to more expressive frameworks with other temporal constraints, besides simple precedence constraints?

Expanding the possibilities of applying plan decoupling to richer temporal frameworks requires us to revisit the computational complexity of its associated problems.

**Research question 2**

For plan decoupling, what are the consequences in terms of the computational complexity of adding different types of temporal constraints to the framework?

A task network with qualitative-temporal constraints (e.g., precedence constraints) can be embedded in a task network with quantitative-temporal constraints (e.g., a simple temporal network). By using this relation, we can compare plan and schedule decoupling and address the following questions.
2.5. RESEARCH QUESTIONS AND THESIS OUTLINE

Research question 3
What is the difference between plan and schedule decoupling (in a quantitative framework)?

If there is a difference between these two preference-preserving approaches, we are interested in the difference in computational complexity.

Research question 4
How does the computational complexity of plan decoupling compare to that of schedule decoupling (in a quantitative framework)?

While the questions above are mostly theoretically motivated, the effects of decoupling on the agents’ performance need to be studied as well. After all, it must be in the agent’s interest to apply a decoupling approach (i.e., the costs should not outweigh the benefits), when the agents are self interested. Here, we make a distinction between the effects of using plan decoupling for all agents together and the effect for an agent individually.

Research question 5
What is the effect of using plan decoupling on the plan-execution costs and plan-construction times for all agents together?

Research question 6
What is the effect of using plan decoupling on the plan-execution costs and plan-construction times for the agents individually?

In the remainder of this thesis, we will address the research questions above as follows. In Chapter 3, we extend the applicability of plan decoupling to a rich qualitative-temporal framework (Research question 1) and study the changes in complexity (Research question 2). In Chapter 4, we consider task networks with quantitative-temporal constraints that extend the task networks with qualitative-temporal constraints. We define plan and schedule decoupling in this quantitative-temporal framework (Research questions 1 and 3), and study their complexity (Research questions 2 and 4). In Chapter 5, we take a more practical point of view and conduct experiments to determine the effect of using plan decoupling on the plan-execution costs and plan-construction times for the agents, both for all agents together (Research question 5) and for an individual agent (Research question 6).

In this thesis, we study coordination problems for self-interested agents that prefer coordination at the lowest possible cost. For this purpose, we will minimise the size of the coordination set (i.e., the number of coordination constraints). It remains to be seen whether the size of the coordination set accurately reflects the additional costs for the agents.
Chapter 3

Decoupling Qualitative Task Networks

In many real-life situations, agents need to solve task-based planning problems. Often, a set of tasks simply needs to be completed without constraints on their completion time (i.e., although earlier completion might be preferred, no deadlines exist). Such situation can be described as a set of tasks that are constrained with qualitative-temporal constraints only.

One such problem that needs to be solved is that of port container transshipment discussed in Section 1.1.2. There, container ships are unloaded by multiple container cranes, which can handle a single container at a time. Because these containers are stacked, a precedence ordering exists on the unloading tasks for these containers (e.g., when container conA is stacked on top of container conB then unload(conA) before unload(conB) must hold). Furthermore, the task sequence for container transshipment (e.g., from the container crane via trucks to the stack) requires synchronisation among the transport agents. For instance, when a container needs to be loaded onto a truck, synchronisation is required between the crane and truck agent for finishing their task together (i.e., craneTask finishes truckTask).

If these agents are self-interested planners, they want to construct plans for completing their sets of tasks independently from the others (i.e., using their own planning tools). In Chapter 2, we have seen that such independent planning gives rise to a coordination problem that needs to be solved. We solve this coordination problem by taking an a-priori coordination-by-design approach, which is called plan decoupling. Here, the idea is to change the original problem instance by adding a set of constraints, called a coordination set, such that the joint plan is guaranteed to be consistent (i.e., the joint plan can be executed without violating any constraint). The advantage of plan decoupling is twofold: (i) it is an acceptable approach for agents at any level of autonomy because it does not require plan revision, and (ii) it constructs an instance-specific coordination set.

We assume that only coordination sets that impose a minimum set of additional constraints are acceptable, when dealing with self-interested agents. For that purpose, we solve the plan-decoupling problem by constructing a coordination set with
The goal of this chapter is to study plan decoupling of a qualitative task network that allows qualitative-temporal constraints to be specified. We represent each task by means of its end points and allow precedence and synchronisation constraints to be defined among the end points of the tasks. In this way, the allowed constraints subsume the basic qualitative-temporal constraints as known from Allen’s interval algebra (Allen, 1983).

Previous work by Valk (2005) studies the cardinal-minimal plan-decoupling problem. However, the framework in which he studied this problem featured only one single type of qualitative-temporal constraint among tasks: sequential ordering ($A$ before $B$). Although this work can be used in problems where only precedence orderings are allowed among the tasks, it is not expressive enough to solve coordination problems for task-based planning problems that allow other qualitative-temporal constraints. Most notably, it lacks the ability to solve the coordination problem when the plans require synchronisation among the agents.

Therefore, the remainder of this chapter is organised as follows. In Section 3.1, we present a framework for describing task networks in which all basic qualitative-temporal constraints can be expressed. We conclude this section with the identification of the plan-coordination problem that can arise in such problem instances. In Section 3.2, we take a plan-decoupling approach to solving this plan-coordination problem for which we define two variants, and study their complexity. These variants of the plan-decoupling problem, in which the planning freedom is reduced in some minimal way, turn out to be intractable. Therefore, we attempt to identify subclasses that are easier to solve from a complexity point of view, and finally formulate a polynomial-time heuristic for constructing a plan decoupling. In Section 3.3, we reflect on the decoupling problem for qualitative networks by looking at the applicability to practical problems.

### 3.1 Qualitative Task Networks

We consider task networks $(\mathcal{T}, \mathcal{C})$ consisting of a set of elementary tasks $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_k\}$ that are constrained by a set of constraints $\mathcal{C}$. One such a framework described in the literature (Buzing et al., 2006; Valk, 2005) consists of a set of ordering constraints between a set of elementary tasks. Such an ordering constraint $\tau$ before $\tau'$ specifies that task $\tau$ must be completed before task $\tau'$ can start. Together, these constraints induce a partial ordering on $\mathcal{T}$.

We represent each task $\tau_i$ as an ordered pair of time-point variables $(t^i_s, t^i_e)$, with $t^i_s$ being the start and $t^i_e$ being the end of the execution of the task $\tau_i$. On the set of time-point variables $T$, we allow a precedence relation $\prec \subseteq T \times T$ and a synchronisation relation $\equiv \subseteq T \times T$ to be defined.

From the relations defined on the time-point variables, we are able to retrieve the qualitative-temporal constraints that are defined on the tasks. In this formalism, all basic qualitative-temporal constraints known from interval algebra (Allen, 1983;
3.1. QUALITATIVE TASK NETWORKS

Dechter, 2003) can be represented, as was shown in Chapter 2.

Note that for every task \( \tau_i \), the associated end points are constrained by \( t^e_i < t^e_i \), and that the relations \( < \) and \( \equiv \) together are assumed to satisfy the following natural properties:

1. \((< \circ \equiv) \subseteq < \) and \((\equiv \circ <) \subseteq < \), that is, \((t < t') \land (t' \equiv t'')\) implies \((t < t'')\), and

2. \((< \cap \equiv) = \emptyset\), that is, \(< \) and \(\equiv\) are orthogonal relations.

Such a network of qualitatively-constrained sets of tasks will be referred to as a qualitative task network, or task network for short. We represent such a qualitative task network as a tuple \( \langle T, <, \equiv \rangle \), where the set of tasks \( T \) is represented by the set of its time-point variables \( T = \{ t^e_\tau, t^c_\tau \mid \tau \in T \} \).

3.1.1 Agents and Task Assignments

We consider a set of agents \( A = \{ A_1, A_2, \ldots, A_n \} \) that are jointly capable of completing the set \( T \) of elementary tasks. Because all tasks are assumed to be elementary (i.e., can be assigned to an agent that is capable of completing it), each task \( \tau \in T \) is assigned to exactly one agent \( A_i \) such that all tasks can be completed.\(^1\) When a task is not elementary, it must be decomposed in a set of elementary tasks and have appropriate precedence and synchronisation constraints set between their end points.

Example 3.1. The non-elementary task of simultaneously lifting a table with two agents can be decomposed in two elementary tasks that both have to start and end at the same time (i.e., decompose task \( \tau \) into elementary tasks \( \tau_1, \tau_2 \) and add the constraint \( \tau_1 \text{ equals } \tau_2 \)). The task \( \tau \) of conducting a certain experiment three times becomes three elementary tasks \( \tau_1, \tau_2, \tau_3 \) without any additional constraints.

Therefore, an assignment of tasks to agents can be conceived as a partitioning \( \{ T_i \}_{i=1}^n \) of the set of tasks \( T \), where \( T_i \) is the set of tasks assigned to agent \( A_i \). As a consequence of this task partitioning, the precedence constraints \(<\) and synchronisation constraints \(\equiv\) are partitioned similarly. For each partition \( T_i \), we get local sets of constraints \( <_i = (\prec \cap (T_i \times T_i)) \) and \( \equiv_i = (\equiv \cap (T_i \times T_i)) \), where \( T_i \) is the set of time-point variables corresponding to the tasks in \( T_i \). We represent such a partition of elementary tasks and its associated constraints as a tuple \( \langle T_i, <_i, \equiv_i \rangle \), where the set of time-point variables \( T_i \) from now on represents the set of tasks \( T_i \).

Note that the original constraints \(<\) and \(\equiv\) are partitioned into the sets of constraints between time-point variables assigned to a single agent, and those constraining time-point variables assigned to different agents. We refer to the former sets of constraints as the intra-agent constraints \(<_i\) and \(\equiv_i\) of agent \( A_i \), respectively, and

\(^1\)How to find a suitable assignment for a set of agents is a separate problem (Shehory and Kraus, 1998; Zlot and Stentz, 2006), and is beyond the scope of this thesis.
refer to the latter set of constraints as the *inter*-agent constraints $\prec_{\text{inter}}, \equiv_{\text{inter}}$. Together, the sets of inter and intra-agent constraints make up the complete sets of constraints, i.e., $\prec = (\prec_{\text{inter}} \cup \bigcup_{i=1}^{n} \prec_{i}, \equiv_{\text{inter}} \cup \bigcup_{i=1}^{n} \equiv_{i})$.

![Figure 3.1: A qualitative task network.](image)

**Example 3.2.** In Figure 3.1, an example is shown of a partitioned and assigned qualitative task network. There are two agents $A_1, A_2$, which have been assigned two tasks each, $\tau_1, \tau_4$ and $\tau_2, \tau_3$, respectively. Each task $\tau_i$ is represented by its start $t_i^s$ and end $t_i^e$ and a precedence ordering $t_i^s \prec t_i^e$ (directed arc) such that the starting time must precede the ending time. There are no further intra-agent constraints defined. There are two inter-agent constraints $\tau_1$ before $\tau_2$ and $\tau_3$ meets $\tau_4$, that are represented by $t_1^e \prec t_2^s$ and $t_3^e \equiv t_4^s$ (semi-dashed edge), respectively.

Instead of representing a partitioned qualitative task network as a tuple $\langle \{T_i\}_{i=1}^{n}, \prec_{\text{inter}} \cup \bigcup_{i=1}^{n} \prec_{i}, \equiv_{\text{inter}} \cup \bigcup_{i=1}^{n} \equiv_{i} \rangle$, we will use the abbreviated notation $\langle \{T_i\}_{i=1}^{n}, \prec_{i}, \equiv_{i} \rangle$ and refer to it as a task network. We will refer to $\langle T_i, \prec_{i}, \equiv_{i} \rangle$ as a sub-task network.

Furthermore, we assume that each agent is aware of all (implied) constraints on its local time-point variables, that can be determined by taking the transitive closure over the sets of constraints. It should hold that $\prec_{i} = (\prec_{i}^+ \cap (T_i \times T_i))$ and similarly for $\equiv_{i}$, where $\rho^+$ is the transitive closure of $\rho$. For a succinct representation of a sub-task network, we take the transitive reduction $\rho^-$ over the sets of local constraints $\prec_{i}$ and $\equiv_{i}$.

### 3.1.2 Agent Plans

In order to complete its sub-task network $\langle T_i, \prec_{i}, \equiv_{i} \rangle$, agent $A_i$ has to construct a plan $\pi_i$ for it which can be executed. Intuitively, such a plan specifies the partial order in which the tasks are to be executed. Therefore, we define an agent’s plan as follows.

**Definition 3.3 (Plan).** A plan $\pi_i$ for a task network $\langle T_i, \prec_{i}, \equiv_{i} \rangle$ is a set of ordering constraints $\langle \pi_i^\prec, \pi_i^\equiv \rangle$ where $\pi_i^\prec$ and $\pi_i^\equiv$ are extensions of $\prec_{i}$ and $\equiv_{i}$, respectively.

**Remark 3.4.** This definition of a plan sets minimal requirements on the planning tools used. The only requirement is that each solution can be represented as a refined...
sub-task network \( \langle T_i, \prec_i, \equiv_i \rangle \), where the relations \( \prec_i \) and \( \equiv_i \) are induced by the constructed solution. Then, we obtain the constructed plan \( \pi_i = \langle \pi_i^\prec, \pi_i^\equiv \rangle = \langle \prec_i^1, \equiv_i^1 \rangle \). Therefore, the plan \( \pi_i \) might be conceived as a qualitative abstraction of the concrete plan constructed.

Example 3.5. Consider the plan for transporting a sea container filled with 12 motorcycles from a factory in Chongqing to the seaport Guangzhou (i.e., task \( \tau_1 \)), and bringing an empty sea container from the seaport to the factory (i.e., task \( \tau_2 \)). The specific planner used for this transportation problem specifies when and where the tasks need to be done, which resources to use, the paperwork that needs to be signed, and by whom. In this framework, we are not interested in the details for executing the tasks, but only in the execution order. Therefore, we use an abstraction of the concrete plan, which in this case might be \( \tau_1 \) before \( \tau_2 \).

A plan that is constructed for a given sub-task network needs to be executable. Because no execution times are specified in the plan \( \pi_i \), a value assignment \( \nu_i \) remains to be constructed that satisfies all ordering constraints specified in the plan. Such a value assignment is called a dispatching (i.e., a dispatch schedule) and can formally be defined as follows.

Definition 3.6 (Dispatching). A dispatching \( \nu_i \) for a task network \( \langle T_i, \prec_i, \equiv_i \rangle \) is a function \( \nu_i : T_i \rightarrow \mathbb{Q} \) such that:

1. \( \forall t, t' \in T_i : t \prec_i t' \implies \nu_i(t) < \nu_i(t') \), and
2. \( \forall t, t' \in T_i : t \equiv_i t' \implies \nu_i(t) = \nu_i(t') \).

Definition 3.7 (Plan Consistency). A plan \( \pi_i = \langle \pi_i^\prec, \pi_i^\equiv \rangle \) for a task network \( \langle T_i, \prec_i, \equiv_i \rangle \) is called consistent if there exists a dispatching \( \nu_i \) for the network \( \langle T_i, \pi_i^\prec, \pi_i^\equiv \rangle \) that results after adding the planning constraints to the original task network.

Similar to planning, an agent \( A_i \) might want to construct a dispatching \( \nu_i \) for its own sub-task network \( \langle T_i, \prec_i, \equiv_i \rangle \). That topic will be discussed in the following chapter.

3.1.3 Plan Coordination

In the previous section, we described the plans that need to be constructed for completing a task network, and the conditions that make these plans consistent. Our approach for solving a given task network is to decompose it into sub-task networks that can be solved by different agents independently. In this section, we define the plan-coordination problem that arises when taking this decomposition approach, and identify the necessary conditions for the consistency of the independently-constructed plans together.

We assume that each local plan \( \pi_i = \langle \pi_i^\prec, \pi_i^\equiv \rangle \) for sub-task network \( \langle T_i, \prec_i, \equiv_i \rangle \) is consistent. The merge of these (locally-consistent) plans together is called the joint
plan \( \pi = \{\pi_i\}_{i=1}^n = \{\{\pi_i^n\}_{i=1}^n, \{\pi_i^{\equiv}\}_{i=1}^n\} \) for task network \( \{\{T_i\}_{i=1}^n, \prec, \equiv\} \). As a result of this joint plan, we get the refined task network \( \{\{T_i\}_{i=1}^n, \prec_{\text{inter}} \cup \{\pi_i^n\}_{i=1}^n, \equiv_{\text{inter}} \cup \{\pi_i^{\equiv}\}_{i=1}^n\} \). Although each local plan \( \pi_i \) is consistent, it is not guaranteed that the joint plan \( \pi \) can be executed without violating any constraint as illustrated by the following example.

![Diagram](image)

Figure 3.2: A plan-uncoordinated qualitative task network with inter-agent precedence constraints.

**Example 3.8.** In Figure 3.2(a), two agents are shown where agents \( A_1 \) and \( A_2 \) have the freedom of planning \( t_2^1 \prec t_1^1 \) and \( t_2^2 \prec t_3^2 \), respectively. But when these plans are joined, a cycle \( \langle t_1^1, t_2^2, t_3^2, t_4^2, t_1^2 \rangle \) is introduced (see Figure 3.2(b)). Such a cycle indicates an inconsistent joint plan, since it implies that \( t_1^1 \) precedes \( t_3^2 \), but also vice versa.

A joint plan needs to be executable, in the sense that a joint dispatching \( \nu = \{\nu_i\}_{i=1}^n \) must exist that satisfies all constraints specified in the independently-constructed plans. This joint dispatching needs to satisfy not only all intra-agent constraints, but also the inter-agent constraints that are not part of any sub-task network.

**Definition 3.9 (Joint Dispatching).** A joint dispatching \( \nu = \{\nu_i\}_{i=1}^n \) for a task network \( \{\{T_i\}_{i=1}^n, \prec, \equiv\} \) is a function \( \nu : T \rightarrow \mathbb{Q} \) such that:

1. \( \nu_i \) is a dispatching for \( \langle T_i, \prec_i, \equiv_i \rangle \),
2. \( \forall t \in T : t \in T_i \) implies \( \nu(t) = \nu_i(t) \),
3. \( \forall t \in T_i, t' \in T_j : t \prec t' \) implies \( \nu(t) < \nu(t') \), and
4. \( \forall t \in T_i, t' \in T_j : t \equiv t' \) implies \( \nu(t) = \nu(t') \).

**Definition 3.10 (Joint Plan Consistency).** A joint plan \( \langle\{\pi_i^n\}_{i=1}^n, \{\pi_i^{\equiv}\}_{i=1}^n\rangle \) for task network \( \{\{T_i\}_{i=1}^n, \prec, \equiv\} \) is called consistent if there exists a joint dispatching \( \nu \) for \( \{\{T_i\}_{i=1}^n, \prec_{\text{inter}} \cup \{\pi_i^n\}_{i=1}^n, \equiv_{\text{inter}} \cup \{\pi_i^{\equiv}\}_{i=1}^n\} \).
Having defined the consistency of a joint plan, we can focus on the construction thereof by a set of agents. Consider the joint plan \( \pi = \{\pi_i\}_{i=1}^n \) for a task network \( \langle \{T_i\}_{i=1}^n, \prec_{\text{inter}} \cup \bigcup_{i=1}^n \prec_{\text{intra}} \cup \bigcup_{i=1}^n \equiv_{\text{intra}} \rangle \). Under the assumption of local plan consistency, all intra-agent constraints are satisfied. Therefore, the question is whether this joint plan also meets all inter-agent constraints \( \prec_{\text{inter}} \) and \( \equiv_{\text{inter}} \). As illustrated by Example 3.8, it is not guaranteed that such a joint plan is consistent, even when all local plans are consistent. Therefore, a coordination problem needs to be solved, which is to guarantee plan coordination (i.e., consistency of the joint plan).

**Definition 3.11** (Plan Coordination). A task network \( \langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle \) is called plan coordinated when the joint plan \( \pi = \{\pi_i\}_{i=1}^n \) is consistent, for every set \( \{\pi_i\}_{i=1}^n \) of locally-consistent plans \( \pi_i \).

The only conflicts that can arise, under the assumption of local plan consistency, in task networks with precedence constraints (i.e., no synchronisation constraints) are directed inter-agent cycles (Buzing et al., 2006; Valk, 2005). Such a cycle might arise due to a combination of constraints in (consistent) local plans and the inter-agent constraints, because the inter-agent constraints are not considered by the agents during planning. If the precedence relation contains a cycle, the partial ordering is broken which makes the plan inconsistent. The task networks we consider, however, can also contain synchronisation constraints, which gives rise to additional potential conflicts that need to be prevented. In fact, the presence of synchronisation constraints might result in a partially-directed cycle.

**Definition 3.12** (Partially-Directed Path). A partially-directed path \( p = (t_1, t_2, \ldots, t_n) \) in a task network \( \langle T, \prec, \equiv \rangle \) is an ordered set of time-point variables such that

1. \( \exists i \in \{1, 2, \ldots, n-1\} : t_i \prec t_{i+1} \), and
2. \( \forall i \in \{1, 2, \ldots, n-1\} : (t_i \prec t_{i+1}) \lor (t_i \equiv t_{i+1}) \).

**Definition 3.13** (Partially-Directed Cycle). A partially-directed cycle \( c = (t_1, t_2, \ldots, t_n, t_{n+1}) \) in a task network \( \langle T, \prec, \equiv \rangle \) is a partially-directed path with \( t_1 = t_{n+1} \).

**Example 3.14.** In Figure 3.3, an example is depicted of two agents with four time-point variables each. There are two pairs of synchronised time-point variables: \( \{t_1^s, t_2^s\} \) and \( \{t_3^s, t_4^s\} \). Note that independent planning might violate these synchronisation constraints even if no directed inter-agent cycles are created. In Figure 3.3(b), time-point variable \( t_1^s \) is planned to occur before \( t_4^s \), while \( t_3^s \) is planned before \( t_2^s \). This clearly violates the given synchronisation constraints \( t_1^s \equiv t_2^s \) and \( t_3^s \equiv t_4^s \), since every valuations satisfying the precedence constraints have \( \nu(t_1^s) < \nu(t_4^s) \) and \( \nu(t_3^s) < \nu(t_2^s) \). However, satisfying the synchronisation constraint \( t_1^s \equiv t_2^s \) requires \( \nu(t_1^s) = \nu(t_2^s) \), which implies \( \nu(t_3^s) < \nu(t_4^s) \) and thereby violates \( t_3^s \equiv t_4^s \).
CHAPTER 3. DECOUPLING QUALITATIVE TASK NETWORKS

Figure 3.3: An uncoordinated qualitative task network with inter-agent synchronisation constraints.

Remark 3.15. Note that these synchronisation constraints cannot simply be interpreted as bidirectional ordering constraints, because cycles consisting only of synchronisation constraints do not form a conflict. For instance, adding the synchronisation constraints $t^1_s \equiv t^4_s$ and $t^2_s \equiv t^3_s$ to the task network of Figure 3.3(a) merely synchronises the starting times of all four tasks.

In summary, we presented a framework for describing qualitative task networks that is richer than the precedence graphs used by Valk (2005). In the defined task networks, a distinction is made between the start and end points of a task. By allowing both precedence and synchronisation constraints, we allow all seven basic qualitative constraints to be specified between the tasks (cf. Valk only allowed ordering of tasks). We showed that the addition of the synchronisations introduces new potential conflicts that need to be dealt with during plan coordination. Therefore, we will study this new plan-coordination framework in the remainder of this chapter.

3.2 Plan Decoupling

In the previous section, a framework was presented for describing qualitative task networks that are assigned to a set of agents. We showed that a plan-coordination problem may arise when these agents construct plans for their sub-task networks, even when all plans are locally consistent.

This plan-coordination problem can be solved by adding constraints $\Gamma^\prec \subseteq \bigcup_{i=1}^n (T_i \times T_i)$ and $\Gamma^\equiv \subseteq \bigcup_{i=1}^n (T_i \times T_i)$ to the sub-task networks that are assigned to the agents. Then, the joint plan is guaranteed to be conflict free, whatever locally-consistent plans the agents construct. In this section, we study this approach called plan decoupling.
3.2. PLAN DECOUPLING

Definition 3.16 (Time Point Depth). In a task network \( \langle T, \prec^*, \equiv^* \rangle \) where \( \prec^* \) is right- and left-closed under composition with \( \equiv^* \) (see Section 3.1), the depth \( \text{depth}(t) \) of a time-point variable \( t \) is defined inductively as follows:

\[
\text{depth}(t) \begin{cases} 
  1 & \text{if } \not\exists t' \in T : t' < t, \\
  1 + \max\{\text{depth}(t') \mid t' < t\} & \text{otherwise}.
\end{cases}
\]

Lemma 3.17. A task network \( \langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle \) is plan coordinated if, and only if, there exists no joint plan \( \pi = \{\pi_i\}_{i=1}^n = \langle \{\pi_i^*\}_{i=1}^n, \{\pi_i^\equiv\}_{i=1}^n \rangle \) of locally-consistent plans \( \pi_i \) for \( \langle T_i, \prec_i, \equiv_i \rangle \), such that the refined task network \( \langle \{T_i\}_{i=1}^n, \prec_{\text{inter}} \cup \{\pi_i^*\}_{i=1}^n, \equiv_{\text{inter}} \cup \{\pi_i^\equiv\}_{i=1}^n \rangle \) contains a partially-directed cycle.

Proof. (\( \Rightarrow \)) By definition, a task network is plan coordinated when, for every joint plan \( \pi = \{\pi_i\}_{i=1}^n \) of locally-consistent plans \( \pi_i \), there exists a joint dispatching \( \nu \). Such a joint dispatching is an assignment \( \nu : T \to \mathbb{Q} \) of values to time-point variables that satisfy all constraints in the refined task network \( \langle \{T_i\}_{i=1}^n, \prec_{\text{inter}} \cup \{\pi_i^*\}_{i=1}^n, \equiv_{\text{inter}} \cup \{\pi_i^\equiv\}_{i=1}^n \rangle \). This immediately excludes the occurrence of any partially-directed cycle \( c = \langle t_1, t_2, \ldots, t_{n+1} \rangle \) where \( t_1 = t_{n+1} = t \), because \( c \) requires \( t_i < t_{i+1} \) to occur at least once, thus requiring \( \nu(t) < \nu(t) \).

(\( \Leftarrow \)) If no joint plan contains a partially-directed cycle, a dispatching \( \nu \) can easily be constructed for the joint plan \( \pi \) by determining the depth of each time-point variable \( t \) in the graph associated with \( \langle \{T_i\}_{i=1}^n, \prec_{\text{inter}} \cup \{\pi_i^*\}_{i=1}^n, \equiv_{\text{inter}} \cup \{\pi_i^\equiv\}_{i=1}^n \rangle \). Since there is no partially-directed cycle, \( \text{depth}(\cdot) \) is well-defined. For each sub-task network \( \langle T_i, \prec_i, \equiv_i \rangle \), the local dispatching \( \nu_i \), defined by \( \nu_i(t) = \text{depth}(t) \) for all \( t \in T_i \), is a dispatching satisfying all local constraints. Clearly, the joint dispatching \( \nu = \{\nu_i\}_{i=1}^n \) is a dispatching satisfying all constraints (i.e., also the inter-agent constraints).

Let us start by defining this notion of a plan decoupling formally, and look at some illustrative examples.

Definition 3.18 (Plan Decoupling). A plan decoupling of a task network \( \langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle \) is formed by adding a set of constraints \( \Gamma = \Gamma^\prec \cup \Gamma^\equiv \) such that the resulting task network \( \langle \{T_i\}_{i=1}^n, \prec \cup \Gamma^\prec \cup \equiv \cup \Gamma^\equiv \rangle \) is plan coordinated.

Example 3.19. In Figure 3.4, we show plan decouplings for task networks for some basic settings with 2 agents, where the dashed arcs are coordination constraints. In Figure 3.4(a), a plan decoupling of the task network is achieved by the coordination set \( \Gamma^\prec = \{t_1^* < t_4^*, t_4^* < t_1^*, t_3^* < t_2^*\} \). Here, a (partially-)directed inter-agent path \( \langle t_3^*, t_4^*, t_1^*, t_2^* \rangle \) exists. Agent 2 cannot extend this into a (partially-)directed cycle by adding \( t_2^* < t_3^* \), because this would make its local plan inconsistent. In Figure 3.4(b), the coordination set \( \Gamma^\prec = \{t_1^* < t_4^*\} \) achieves a plan decoupling, because—without the constraints in \( \Gamma^\prec \)—a partially-directed inter-agent path through agent 1 exists with a consistent local plan \( \pi_1 \). In Figures 3.4(c) and 3.4(d), similar remarks can be made, despite the fact that these task networks differ slightly from the previous two. Clearly, no inter-agent planning conflicts can occur in any of these instances after adding the constraints in \( \Gamma^\prec \).
Figure 3.4: Example plan decouplings.

Although a plan decoupling might result directly from task allocation, in general, a task network needs to be changed into a plan decoupling by adding a set $\Gamma = \Gamma_\prec \cup \Gamma_\equiv$. Such a set of additional constraints is called a *coordination set*. First, we need to address the question whether it is always possible to make this change by adding a set of constraints. It is not difficult to show that for every task network there exists a coordination set that changes it into a plan decoupling when added to it, which consists of *intra*-agent precedence and/or synchronisation constraints. The idea is to extend the set of local constraints to form a total ordering on the time-point variables $T_i$ in such a way that their union, together with the inter-agent constraints, constitutes a partial order (i.e., does not contain a partially-directed cycle).

**Proposition 3.20.** For every consistent task network $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$, there exists a coordination set $\Gamma = \Gamma_\prec \cup \Gamma_\equiv$ such that $\langle \{T_i\}_{i=1}^n, \prec \cup \Gamma_\prec, \equiv \cup \Gamma_\equiv \rangle$ forms a plan decoupling.

**Proof.** Consider a task network $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$. Since $\prec \cup \equiv$ is acyclic, there exists a total order $\prec^* \cup \equiv^*$ on the set of time-point variables $\{T_i\}_{i=1}^n$, where $\prec^*$ and $\equiv^*$ are extensions of $\prec$ and $\equiv$, respectively.

Consider the refined sub-task networks $\langle T_i, \prec_i^*, \equiv_i^* \rangle$, where $\prec_i^* = (\prec^* \cap (T_i \times T_i))$ and $\equiv_i^* = (\equiv^* \cap (T_i \times T_i))$. Clearly, $\prec_i^*$ and $\equiv_i^*$ are extensions of $\prec_i$ and $\equiv_i$, respectively.
respectively, and \( \prec_i \cup \equiv_i \) is a total-order relation on \( T_i \). Because no further refinements of these sub-task networks are possible (it is a total order), each agent \( A_i \) constructs the locally-consistent plan \( \pi_i = \langle \prec_i, \equiv_i \rangle \). Thus, the joint plan \( \pi = \langle \bigcup_{i=1}^n \prec_i, \bigcup_{i=1}^n \equiv_i \rangle \) is consistent.

The coordination set \( \Gamma = \Gamma^\prec \cup \Gamma^\equiv \) can then be constructed as \( \Gamma = \bigcup_{i=1}^n \Gamma^\prec_i \cup \bigcup_{i=1}^n \Gamma^\equiv_i \), with \( \Gamma^\prec_i = \prec_i \setminus \prec_i \) and \( \Gamma^\equiv_i = \equiv_i \setminus \equiv_i \).

Now we can define the Plan-Decoupling Problem (PDP) of finding such a coordination set:

**Plan-Decoupling Problem (PDP)**

Find a coordination set \( \Gamma = \bigcup_{i=1}^n \Gamma^\prec_i \cup \bigcup_{i=1}^n \Gamma^\equiv_i \) for a task network \( \langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle \), such that \( \langle \{T_i\}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv \rangle \) is plan coordinated.

Note here that this problem is to construct an arbitrary coordination set, without setting any quality guarantees of the constructed solution (i.e., the coordination set is of arbitrary size). As a direct consequence of the construction used in the proof of Proposition 3.20, constructing such a coordination set can be done in polynomial time.

**Corollary 3.21.** PDP can be solved in polynomial time.

Instead of constructing a coordination set of arbitrary quality, we are also interested in constructing one with some desirable property. Therefore, consider the plan decouplings in Figure 3.4. In the Figures 3.4(a) and 3.4(b), two plan decouplings are shown for the same task network, with the coordination sets containing 3 and 1 constraints, respectively. Clearly, self-interested agents will prefer the plan-decoupling problem to be solved with the latter coordination set.

Recall that the agents want to become plan coordinated at a lowest cost (i.e., a minimal reduction in the planning freedom of the agents). Because this notion of minimality can be defined as a local or a global property, we define two variants of the PDP with a minimal reduction in the planning freedom. The first variant is to find a subset-minimal coordination set that, from a local perspective, reduces the planning freedom in a minimal way (i.e., removing any constraint from any coordination subset results in the task network being plan uncoordinated):

**Subset-Minimal Plan-Decoupling Problem (sMPDP)**

Find a subset-minimal coordination set \( \Gamma = \bigcup_{i=1}^n \Gamma^\prec_i \cup \bigcup_{i=1}^n \Gamma^\equiv_i \) for a task network \( \langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle \), such that \( \langle \{T_i\}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv \rangle \) is plan coordinated.

Because this problem does not guarantee a globally-minimal coordination set, we define a second variant which is to find a cardinal-minimal coordination set:

**Cardinal-Minimal Plan-Decoupling Problem (cMPDP)**

Find a cardinal-minimal coordination set \( \Gamma = \bigcup_{i=1}^n \Gamma^\prec_i \cup \bigcup_{i=1}^n \Gamma^\equiv_i \) for a task network \( \langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle \), such that \( \langle \{T_i\}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv \rangle \) is plan coordinated.
3.2.1 Complexity

In this section, we study the complexity of three decision problems related to plan decoupling. First, we will determine the complexity of the Plan-Decoupling Recognition Problem (PDRP), which decides whether or not a set of constraints results in a plan decoupling when added to the task network. We start with this problem, because it is the basic problem that needs to be solved by the other two plan-decoupling problems. Second, we study the complexity of the decision variants of the subset-minimal, and, finally, the cardinal-minimal plan-decoupling problem defined above.

Plan-Decoupling Recognition Problem (PDRP)

INSTANCE: A task network \( \langle \{ T_i \}_{i=1}^n, \prec, \equiv \rangle \) and a set of constraints \( \Gamma = \Gamma^\prec \cup \Gamma^\equiv \).

QUESTION: Is \( \langle \{ T_i \}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv \rangle \) plan coordinated?

Proposition 3.22. PDRP is \( \text{coNP-complete} \).

Proof. Membership is proven by the fact that a no-certificate can be verified in polynomial time. Recall that verifying whether a task network is plan coordinated can be accomplished by verifying the condition stated in Lemma 3.17. Thus, such a no-certificate consists of a set of locally-consistent plans \( \{ \pi_i \}_{i=1}^n = \{ \{ \pi_i^\prec \}_{i=1}^n, \{ \pi_i^\equiv \}_{i=1}^n \} \) for the sub-task networks in task network \( \langle \{ T_i \}_{i=1}^n, \prec \cup \bigcup_{i=1}^n \pi_i^\prec, \equiv \cup \bigcup_{i=1}^n \pi_i^\equiv \rangle \) such that \( \langle \{ T_i \}_{i=1}^n, \prec \cup \bigcup_{i=1}^n \pi_i^\prec, \equiv \cup \bigcup_{i=1}^n \pi_i^\equiv \rangle \) contains a partially-directed cycle. Verification can be done in polynomial time by applying Topological-Sort (Cormen et al., 1990) to verify local consistency of each plan \( \pi_i \), and to verify the existence of a partially-directed cycle in \( \{ \pi_i \}_{i=1}^n \).

Hardness for this class can be proven by the fact that the PDRP for task networks with only precedence constraints is subsumed. Because this contained problem is known to be coNP-complete (Valk, 2005), it follows that the more general PDRP is coNP-hard.

Now we turn to the plan-decoupling problem itself, for which we study the subset-minimal and cardinal-minimal variant. We start with the subset-minimal plan-decoupling problem, for which we define the decision variant as follows.

Decision variant of sMPDP

INSTANCE: A task network \( \langle \{ T_i \}_{i=1}^n, \prec, \equiv \rangle \), a coordination set \( \hat{\Gamma} = \hat{\Gamma}^\prec \cup \hat{\Gamma}^\equiv \), and a positive integer \( K \).

QUESTION: Does there exist a coordination set \( \Gamma = \Gamma^\prec \cup \Gamma^\equiv \), with \( \Gamma^\prec \subseteq \hat{\Gamma}^\prec \), \( \Gamma^\equiv \subseteq \hat{\Gamma}^\equiv \), and \( |\Gamma| \leq K \), such that \( \langle \{ T_i \}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv \rangle \) is plan coordinated?

Based on Proposition 3.22 and Corollary 3.21, we get the following two propositions on the complexity of sMPDP.
Proposition 3.23. **sMPDP is coNP-hard.**

*Proof.* Hardness is proven by the fact that for $K = 0$, the sMPDP reduces to the PDRP.

Proposition 3.24. **sMPDP $\in \Delta^p_2$.**

*Proof.* Membership is shown by the fact that a coordination set can be constructed in polynomial time (see Corollary 3.21). Then iteratively check for every element in this coordination set whether it can be removed without the resulting task network becoming plan uncoordinated using a PDRP oracle. Because the coordination set has a size polynomial in the input, a PDRP-oracle needs to be consulted a polynomial number of times. Due to the PDRP-oracle being coNP-complete (see Proposition 3.22), the corollary follows.

Unfortunately, this leaves open the exact complexity of sMPDP.

Next, we study the complexity of the cardinal-minimal plan-decoupling problem. To determine the complexity of the cMPDP we formulate the following decision variant.

**Decision variant of cMPDP**

**INSTANCE:** A task network $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ and a positive integer $K$.

**QUESTION:** Does there exist a coordination set $\Gamma = \Gamma^\prec \cup \Gamma^\equiv = \bigcup_{i=1}^n \Gamma^\prec_i \cup \bigcup_{i=1}^n \Gamma^\equiv_i$ with $|\Gamma| \leq K$, such that $\langle \{T_i\}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv \rangle$ is plan coordinated?

Proposition 3.25. **cMPDP is $\Sigma^p_2$-complete.**

*Proof.* Membership is proven by the fact that a yes-certificate can be verified in nondeterministic polynomial time. After guessing a coordination set $\Gamma = \bigcup_{i=1}^n \Gamma^\prec_i \cup \bigcup_{i=1}^n \Gamma^\equiv_i$, verifying whether $\langle \{T_i\}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv \rangle$ is plan coordinated is exactly the PDRP (see Proposition 3.22). Because checking whether $|\Gamma| \leq K$ can be done polynomial time, we have that cMPDP $\in \text{NP}^{\text{coNP}} = \Sigma^p_2$.

Hardness for this class can be proven by the fact that the cMPDP for task networks with only precedence constraints is subsumed. Because this contained problem is known to be $\Sigma^p_2$-complete (Valk, 2005), it follows that the more general cMPDP is $\Sigma^p_2$-hard.

3.2.2 **Factors Influencing the Complexity of Plan-Decoupling Problems**

The complexity of different problems related to plan decoupling shows that we are left with highly-intractable problems, in general. Because such complexity results are based on the hardest problem instances, it might well be that—possibly large—subclasses exist for which the complexity is lower.
All plan-decoupling problems take a task network as input. The (maximum) size of such a task network \( \langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle \) depends on the number of agents and the number of tasks per agent. Note that the number of (allowed) precedence and synchronisation constraints depends on these variables. Therefore, the number of tasks per agent and the number of agents are expected to be sources of complexity. In the following paragraphs, we study the complexity of some subclasses of plan-decoupling problems where either the number of tasks per agent, or the number of agents is limited.

**Limited Number of Tasks per Agent**  It seems reasonable to assume that one source of complexity is the number of tasks assigned to each agent, and the complexity of the single-agent planning problems that results. However, the proof for the coNP-completeness of PDRP uses task networks with at most 2 tasks per agent (i.e., \( \forall i : |T_i| \leq 2 \)).

**Corollary 3.26.** PDRP with \( \forall i : |T_i| \leq c \), with \( c \geq 2 \), is coNP-complete.

**Proof.** Membership is proven by the fact that PDRP is coNP-complete (see Proposition 3.22).

Hardness is due to the construction used in the proof for coNP-hardness of verifying whether a task network with only precedence constraints is plan coordinated (see Theorem 4.8, pp. 78 (Valk, 2005)). In that reduction, task networks \( \langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle \) with \( \equiv = \emptyset \) are used with \( \forall i : |T_i| \leq 2 \). □

Additionally, task networks with at most 4 tasks per agent (i.e., \( \forall i : |T_i| \leq 4 \)) are used in the proof of the \( \Sigma^p_2 \)-completeness of cMPDP.

**Corollary 3.27.** cMPDP with \( \forall i : |T_i| \leq c \), with \( c \geq 4 \), is \( \Sigma^p_2 \)-complete.

**Proof.** Membership is proven by the fact that cMPDP is \( \Sigma^p_2 \)-complete (see Proposition 3.25).

Hardness is due to the reduction used in the proof for \( \Sigma^p_2 \)-hardness of deciding whether a coordination set of size at most \( K \) exists for a task network with only precedence constraints (see Theorem 4.13, pp. 81 (Valk, 2005)). In that reduction, task networks \( \langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle \) with \( \equiv = \emptyset \) are used with \( \forall i : |T_i| \leq 4 \).

Hence, the problem remains difficult even with few number of tasks per agents. In Table 3.1, we summarise the complexity results for these subclasses. Notice that it remains an open problem whether cMPDP is \( \Sigma^p_2 \)-hard for task network with at most 2, or at most 3 tasks assigned to each agent.

**Limited Number of Agents**  Another source of complexity might be the number of agents involved. Indeed, if we limit the number of agents, it can be shown that the PDRP is in P for any fixed number of agents. This can be proven by reducing the problem to inter-agent cycle-testing, which can be done in polynomial time (Cormen et al., 1990).
| $|T_i|$ | $\leq 2$ | $|T_i| \geq 4$ |
|---|---|---|
| PDRP | coNP-complete | coNP-complete |
| cMPDP | $\Sigma_2^p$-complete | $\Sigma_2^p$-complete |

Table 3.1: Complexity of PDRP and cMPDP with limited number of tasks per agent.

**Definition 3.28** (Simple Oriented Cycle). A sequence of arcs $\langle (v_1, v_2), \ldots, (v_n, v_{n+1}) \rangle$ in a directed graph $G = \langle V, A \rangle$ is called a simple oriented cycle of length $n$ if all $(v_i, v_{i+1}) \in A$, all $v_1, v_2, \ldots, v_n$ are distinct, and $v_1 = v_{n+1}$.

**Lemma 3.29.** PDRP with fixed $|A| \in \mathbb{P}$.

*Proof.* Consider the translation of a task network $\langle \{T_i\}_{i=1}^\alpha, \prec \rangle$ with a fixed number of agents $|A| = \alpha$ to a directed graph $G = \langle V, A \rangle$ with $V = \bigcup_{i=1}^\alpha V_i$ defined as follows.

1. $V_i = \{ v \mid t \in T_i \}$
2. $A = (V_i \times V_i) \setminus \{(v_k, v_j) \mid t_j \prec t_k \in \prec^+\} \cup \{(v_j, v_k) \mid t_j \prec t_k \in \prec\}$

It is clear that the above transformation gives a directed graph and a partitioning of its vertices. Note that no simple oriented cycles are formed from the transitive closure $\prec^+$, but that such cycles are formed inside the partitions.

A no-certificate for PDRP on a task network $\langle \{T_i\}_{i=1}^\alpha, \prec \rangle$ with fixed $|A| = \alpha$ is a set of locally-consistent plans $\{\pi_i\}_{i=1}^\alpha = \{\langle T_i, \prec_i^\alpha \rangle\}_{i=1}^\alpha$ such that $\langle \{T_i\}_{i=1}^\alpha, \prec_{\text{inter}} \cup \bigcup_{i=1}^\alpha \prec_i^\alpha \rangle$ contains a partially-directed cycle. Clearly, this cycle passes through at least 2 and at most $\alpha$ partitions, because each plan $\pi_i$ is consistent (i.e., does not contain a partially-directed cycle).

Such a no-certificate implies the existence of a simple oriented cycle in $G = \langle V, A \rangle$ of length at most $2\alpha$. Moreover, a cycle contains at least 2 arcs $(t, u), (v, w)$ where vertices $u, v$ belong to one partition and $t, w$ do not, possibly with $u = v$. Additionally, a cycle needs at most 2 vertices from each partition and needs each partition at most once, because $(u, v)$ and $(v, u)$ cannot both be arcs in an acyclic partition.

A simple oriented cycle $c = \langle (v_1, v_2), \ldots, (v_{\alpha}, v_{\alpha+1}) \rangle$ of length $2\alpha$ in $G$ maps to a no-certificate for PDRP on a task network $\langle \{T_i\}_{i=1}^\alpha, \prec \rangle$ with fixed $|A| = \alpha$ by: $\pi_i = \langle T_i, \prec \cup \{t_j \prec t_k \mid v_j, v_k \in V_i \wedge (v_j, v_k) \in c\} \rangle$

Hence, the PDRP problem equals the checking of the existence of a simple oriented cycle of length at most $2\alpha$, which can be done in polynomial time. $\square$
As a consequence, the associated cMPDP-problems are in **NP**. The cMPDP are **NP**-complete for all fixed values \(|A| \geq 3\), which can be proven by reduction from **3-Partite Vertex Cover** (Poljak, 1974).²

### 3-Partite Vertex Cover

**INSTANCE:** Graph \( G = \langle V_1 \cup V_2 \cup V_3, E \rangle \) and a positive integer \( K \leq |V_1 \cup V_2 \cup V_3| \).

**QUESTION:** Is there a subset \( V' \subseteq V_1 \cup V_2 \cup V_3 \) with \(|V'| \leq K\) such that for each edge \( \{v_i, v_j\} \in E \) at least one of \( v_i \) and \( v_j \) belongs to \( V' \)?

**Lemma 3.30.** **3-Partite Vertex Cover** \( \propto \) **cMPDP** with fixed \(|A| = 3\).

**Proof.** Consider the following reduction from a **3-Partite Vertex Cover** instance \( G = \langle V_1 \cup V_2 \cup V_3, E \rangle \) to a task network with 3 agents \( \langle \{T_i\}_{i=1}^3, \prec \rangle \).

1. \( T_i = T_{i,1} \cup T_{i,2} \), with
   (a) \( T_{i,1} = \{ t_1 \mid v \in V_i \} \)
   (b) \( T_{i,2} = \{ t_2 \mid v \in V_i \} \)

2. \( \prec = \prec_{\text{inter}} \cup \bigcup_{i=1}^3 \prec_i \)
   (a) \( \prec_{\text{inter}} = \{ u_1 \prec v_2, v_1 \prec u_2 \mid \{u, v\} \in E \} \)
   (b) \( \prec_i = (T_{i,1} \times T_{i,2}) \setminus \{ (t_1, t_2) \mid t \in V_i \} \)

It is clear that the above transformation results in a task network with 3 agents.

A yes-certificate for **3-Partite Vertex Cover** on a graph \( G = \langle V_1 \cup V_2 \cup V_3, E \rangle \) and a positive integer \( K \) is a vertex cover \( V' \subseteq V_1 \cup V_2 \cup V_3 \) with \(|V'| \leq K\) such that for all edges \( \{v_i, v_j\} \in E \) it holds that \( v_i \in V' \) or \( v_j \in V' \).

A yes-certificate for **cMPDP** on a task network \( \langle \{T_i\}_{i=1}^3, \prec \rangle \) with fixed \( n = |A| = 3 \) and a positive integer \( K \), is a coordination set \( \Gamma \subseteq \bigcup_{i=1}^3 (T_i \times T_i) \) with \(|\Gamma| \leq K\) such that \( \langle \{T_i\}_{i=1}^3, \prec \cup \Gamma \rangle \) is plan coordinated.

A yes-certificate for **3-Partite Vertex Cover** is mapped to a yes-certificate for **cMPDP** with fixed \(|A| = 3\) by: \( \Gamma = \{ t_1 \prec t_2 \mid v \in V' \} \). It is easy to see that every selected vertex represents an arc that breaks such a cycle.

A yes-certificate for **cMPDP** on a task network \( \langle \{T_i\}_{i=1}^n, \prec \rangle \) with fixed \(|A| = 3\) is mapped to a yes-certificate for **3-Partite Vertex Cover** by: \( V' = \{ v \mid t_1 \prec t_2 \in \Gamma \} \). It is easy to see that \( V' \) is a vertex cover, because every edge represents a potential cycle and every selected vertex represents an arc that breaks such a cycle.

²The result that **3-Partite Vertex Cover** is **NP**-complete is due to the construction of the reduction given by Poljak (1974) from **Vertex Cover** (Garey and Johnson, 1979).
3.2. PLAN DECOUPLING

Proposition 3.31. cMPDP with fixed $|\mathcal{A}| = c$, with $c \geq 3$, is NP-complete.

Proof. Membership is proven by the fact that a yes-certificate of an cMPDP on a task network $\langle \{T_i\}_{i=1}^n, \prec \rangle$ with fixed $|\mathcal{A}| \geq 3$ instance can be verified in polynomial time (see Lemma 3.29).

Hardness for this class follows from the fact that there exists a polynomial-time reduction from the NP-complete 3-PARTITE VERTEX COVER problem to this problem (see Lemma 3.30).

In Table 3.2, we summarise the complexity results for the subclasses of PDRP and cMPDP with limited number of agents. Notice that the hardness of cMPDP with $|\mathcal{A}| = 2$ remains an open question.

| $|\mathcal{A}|$ | PDRP | cMPDP |
|----------------|------|-------|
| 2              | P    | NP-complete |
| $\geq 3$       | P    |       |

Table 3.2: Complexity of PDRP and cMPDP with limited number of agents.

From this section, we can conclude that PDRP and cMPDP are intractable even when the problem instances are limited to consist of few agents, or tasks per agents.

3.2.3 A Heuristic for Constructing Coordination Sets

Because most subclasses of the plan-decoupling problems are intractable (see Section 3.2.2), we turn to approximation algorithms. However, from previous work (Valk, 2005), we know that the approximability of cMPDP for tasks networks with only precedence constraints is APX-hard. This means that it is unlikely that there exists a polynomial-time algorithm for it with a performance ratio that is bounded by a constant. Nevertheless, coordination sets need to be constructable in polynomial time in order to guarantee timely plan coordination. In this section, we formulate a polynomial-time heuristic for constructing coordination sets.

As far as we know, it is currently the best heuristic for constructing coordination sets for the plan-decoupling problem.

A Depth-Partitioning Heuristic

Now, we will describe a depth-partitioning algorithm for constructing a coordination set. This coordination set is sufficient, but not necessarily minimal. This algorithm constructs such a coordination set by assigning a depth value $\text{depth}(t)$ to each time-point variable, and adding precedence constraints between each pair of variables (belonging to the same agent) with a different depth value. Two time-point variables with a precedence constraint between them $t \prec t'$ must have different depth values $\text{depth}(t) < \text{depth}(t')$, while two synchronised time-point variables $t \equiv t'$ must have
Algorithm 1 Depth-Partitioning Algorithm.

1. Take the task \(\langle\{T_i\}_{i=1}^n, \prec, \equiv\rangle\) and construct the depth partitions \(T^d_i = \{t \in \bigcup_{i=1}^n T_i \mid \text{depth}(t) = d\}\) of time-point variables having the same depth. Here, the depth \(\text{depth}(t)\) is defined as: If \(t\) does not have any predecessors in \(\prec\) then \(\text{depth}(t) = 1\), else \(\text{depth}(t) = \max(1 + \max\{\text{depth}(t') \mid t' \prec t\}, \max\{\text{depth}(t') \mid t' \equiv t\})\).

2. For all \(i, d\), construct the partitions \(T^d_i = T^d \cap T_i\).

3. For all \(i\), construct the sequence \(\langle T^{d_1}_i, T^{d_2}_i, \ldots, T^{d_k}_i\rangle\) of all (non-empty) partitions \(T^{d_j}_i\) of \(T_i\) sorted in increasing values of the depth value \(d_j\).

4. For all \(i\), construct \(\Gamma^{\prec}_i = \bigcup_{j=1}^{k-1} (T^{d_j}_i \times T^{d_{j+1}}_i) \setminus \prec_i\) (i.e., add a precedence constraint between each pair of time-point variables from subsequent depth partitions, except those that were already constrained).

5. For all \(i\), construct \(\Gamma^{\equiv}_i = \bigcup_{j=1}^k (T^{d_j}_i \times T^{\equiv, d_j}_i) \setminus \equiv_i\) with \(T^{\equiv, d_j}_i = \{t \in T^{d_j}_i \mid \exists t' \not\in T_i : t \equiv t'\}\) (i.e., add a synchronisation constraint between each pair of time-point variable from the same depth partition that are involved in an inter-agent synchronisation constraint, except those that were already constrained).

6. Return \(\Gamma = \bigcup_{i=1}^n \Gamma^{\prec}_i \cup \bigcup_{i=1}^n \Gamma^{\equiv}_i\).

the same depth value \(\text{depth}(t) = \text{depth}(t')\). These relations are required by the inclusion properties of the composition of \(\prec\) and \(\equiv\) (see Section 3.1). We then partition the time-point variables based on their depth values into one partition for each depth value. By adding precedence constraints between the time-point variables from different partitions, all potential partially-directed cycles traversing through precedence constraints are prevented. Although this prevents all partially-directed inter-agent cycles through different partitions, potential conflicts remain that involve synchronisation constraints only. These remaining conflicts must reside within each depth partition (i.e., the involved time-point variables must have the same depth value). To prevent such conflicts, all time-point variables involved in a synchronisation constraint in the same partition are synchronised. This depth-partitioning algorithm for constructing a coordination set can be described as shown in Algorithm 1.

Example 3.32. Applying Algorithm 1 to the plan-uncoordinated qualitative task network depicted in Figure 3.3(a) gives the following. In step 1, the algorithm determines the depth value of each time-point variable, and partitions the time-point variables based on these depth value into the sets \(T^1 = \{t_1^s, t_2^s, t_3^s, t_4^s\}\) and \(T^2 = \{t_1^e, t_2^e, t_3^e, t_4^e\}\). In steps 2 and 3, the algorithm partitions each partition of time-point variables based on the agents they belong to, and orders all non-empty sets with increasing depth value. For agent \(A_1\), these steps result in \(\langle\{t_1^s, t_4^s\}, \{t_1^e, t_4^e\}\rangle\). In
3.2. PLAN DECOUPLING

step 4, a precedence constraint is added between a pair of time-point variables from subsequent depth partitions when no such constraint exists in the task network. For agent $A_1$, this results in the set $\Gamma_1^\succ = \{ t_1^* < t_4^*, t_4^* < t_1^* \}$ (i.e., the dashed arcs). In step 5, the algorithm adds a synchronisation constraint between each pair of time-point variables in the same depth partition and that are involved in an inter-agent synchronisation constraint, except for those constraints that already exist in the task network. For agent $A_1$, this results in the set $\Gamma_1^\equiv = \{ t_1^* \equiv t_4^* \}$ (i.e., the dashed edge). Adding these constraints results in the situation depicted in Figure 3.5.

Figure 3.5: Plan decoupling two agents by depth partitioning.

**Remark 3.33.** In Example 3.32, using only those time-point variables that are involved in an inter-agent constraint (i.e., $t_1^*, t_2^*, t_3^*, t_4^*$), the algorithm would have constructed the coordination sets $\Gamma_1^\equiv = \{ t_1^* \equiv t_4^* \}$ and $\Gamma_2^\equiv = \{ t_2^* \equiv t_3^* \}$, which also results in the task network becoming plan coordinated.

In the following proposition, it is proven that generating a coordination set based on this algorithm results in a sufficient—not necessarily minimal—coordination set $\Gamma$.

**Proposition 3.34.** Given a task network $\langle \{ T_i \}_{i=1}^n, , \prec, \equiv \rangle$. Let $\Gamma = \Gamma^\prec \cup \Gamma^\equiv$ be the coordination set constructed according to Algorithm 1. Then the resulting $\langle \{ T_i \}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv \rangle$ is plan coordinated.

**Proof.** For the sake of contradiction, suppose that the resulting task network $\langle \{ T_i \}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv \rangle$ does not form a plan decoupling. Then, by Lemma 3.17, some partially-directed cycle $c = \langle t_1, t_2, \ldots, t_m, t_1 \rangle$ must exist. Such a cycle consists of subsequences of time-point variables belonging to a single agent (intra-agent paths) and subsequences of time-point variables belonging to different agents (inter-agent paths).

First, note that, due to the construction of every set $\Gamma_i$, for every subsequence $\langle t_h, t_{h+1} \rangle$ of an intra-agent path $\langle t_j, \ldots, t_k \rangle$ of agent $A_i$, it must hold that

$$\text{depth}(t_h) \leq \text{depth}(t_{h+1}). \quad (3.1)$$

Otherwise, $\text{depth}(t_h) > \text{depth}(t_{h+1})$ would imply that $t_{h+1} < t_h \in \Gamma_i^\succ$, and, therefore, together with either the planned $t_h \prec^* t_{h+1}$ or $t_h \equiv^* t_{h+1}$ would create a local partially-directed cycle.
CHAPTER 3. DECOUPLING QUALITATIVE TASK NETWORKS

Second, for every inter-agent subsequence \(⟨t_h, t_{h+1}⟩\) of the partially-directed cycle \(c\), it must hold that either \(t_h \prec t_{h+1}\) or \(t_h \equiv t_{h+1}\). According to Algorithm 1, this must imply that \(\text{depth}(t_h) = \text{depth}(t_{h+1})\) or \(\text{depth}(t_h) < \text{depth}(t_{h+1})\), respectively.

Hence, we conclude that the depths \(\text{depth}(t_i)\) of the time points \(t_i\) in the partially-directed cycle \(c\) starting with \(\text{depth}(t_1)\) constitute a monotonically non-decreasing sequence. But this immediately implies that \(\text{depth}(t_1) = \text{depth}(t_2) = \cdots = \text{depth}(t_m)\). Therefore, since \(c\) is a partially-directed cycle, there must exist a subsequence \(⟨t_h, t_{h+1}⟩\) such that either \(t_h \prec_{\text{inter}} t_{h+1}\) or \(t_h \prec^* t_{h+1}\) such that \(\text{depth}(t_h) = \text{depth}(t_{h+1})\). Clearly, \(t_h \prec_{\text{inter}} t_{h+1}\) implies \(\text{depth}(t_h) < \text{depth}(t_{h+1})\) since \(\prec_{\text{inter}} \subseteq \prec\). Hence, there exists an agent plan \(π_i\) containing a tuple \(t_h \prec^* t_{h+1}\) such that \(\text{depth}(t_h) = \text{depth}(t_{h+1})\). Let \(⟨t_j, \ldots, t_h, t_{h+1}, \ldots, t_{j'}⟩\) be the maximal intra-agent path containing both \(t_h\) and \(t_{h+1}\). Then, by construction of the set \(Γ_i\), this implies that not both \(t_j\) and \(t_{j'}\) occur in \(T_i^{\equiv k}\) for any \(k\), since this would imply that the depth of all time-point variables in between them would be equal to \(k\) preventing \(t_h \prec^* t_{h+1}\). Hence, at least \(t_j\) has a \(\prec\)-predecessor \(t'_{j'}\) or \(t_{j'}\) has a \(\prec\)-successor \(t'_{j'}\). In both cases, \(c\) contains an occurrence of an inter-agent constraint \(\prec_{\text{inter}}\) implying that either \(\text{depth}(t'_{j'}) < \text{depth}(t_j)\) or \(\text{depth}(t_{j'}) < \text{depth}(t'_{j'})\), contradicting the fact that \(\text{depth}(t) = \text{depth}(t')\) should hold for every pair \(t, t'\) occurring in the cycle. Hence, such a partially-directed cycle cannot exist, and, therefore, the resulting task is plan coordinated.

Before constructing a coordination set for the problem instance, we can do some preprocessing to remove parts of the task network that cannot be part of any conflict requiring coordination. For instance, all those agents that cannot extend their sets of local constraints such that a partially-directed inter-agent cycle might result can be discarded. Besides agents that are not involved in any inter-agent constraints, this class includes two other types of agents: (i) those with only outgoing or only incoming precedence constraints, and (ii) those with only exactly one inter-agent synchronisation constraint. Note that removing these agents together with their inter-agent constraints might result in a new situation where other agents can be excluded, because they cannot construct part of a directed inter-agent cycle. Hence, removing such agents should be done iteratively until no agents can be removed again.

Furthermore, some time-point variables might be used to increase the length of a partially-directed inter-agent cycle. However, when the smaller cycle is prevented, it follows that the larger inter-agent cycle is also prevent. Hence, for achieving plan coordination, only those time-point variables need to be considered that are involved in an inter-agent constraint.

**Example 3.35.** Consider the task network depicted in Figure 3.6. Here, four agents are shown with multiple inter-agent constraints between them. However, agent \(A_4\) only has incoming precedence constraints, and does, therefore, not need to be considered. But then agent \(A_3\) can also be removed because it now has only incoming constraints. Therefore, there only is a real plan-coordination problem between agents \(A_1\) and \(A_2\).
Let us define a condensed task network as the network where the above, and possibly other, irrelevant parts are removed. Then a coordination set for the condensed task network also is a solution for that problem for the original task network. Note that this analysis is not meant to be exhaustive, nor did we strive to achieve this. Moreover, these remarks do not change any of the general complexity results above, but are more of a practical use.

3.3 Discussion

In this chapter, we studied plan decoupling of a set of tasks constrained by qualitative-temporal constraints, called a qualitative task network. First, we defined these qualitative task networks for representing an assigned set of primitive tasks. We represent each task by means of its start and end point, and allow these time-point variables to be constrained with precedence and synchronisation constraints.

In these qualitative task networks, we identified the plan-coordination problem that arises when allowing all agents to construct locally-consistent plans. We defined two variants of the plan-decoupling problem for constructing either a subset-minimal or a cardinal-minimal coordination set. In this framework, we gave proofs for the computational complexity of the plan-decoupling problems. It turned out that the decision problems for both variants are intractable in general. We showed that the complexity of the problems does not increase when also allowing synchronisation constraints compared with allowing precedence constraints only (see previous work by Valk (2005)). We also studied the complexity of subclasses where the number of agents or the number of tasks per agent is limited. These subclasses were shown to remain intractable. Finally, we were able to formulate a polynomial-time heuristic for constructing a plan decoupling, for which we provided a proof of correctness.

Plan decoupling guarantees the existence of a joint dispatching for executing the joint plan. However, it does not give any guarantees with respect to the quality of these joint dispatchings, nor on the quality of the dispatchings of the constructed local plans.

For future work, more involved variants of the plan-decoupling problems should be studied that guarantee other system properties besides plan coordination. For
instance, a coordinator might want to coordinate his agents in such a way that a joint dispatching is not only guaranteed to exist, but is guaranteed to have a minimum makespan (i.e., the time required for executing the plan). Such a variant is likely to be important for practical applicability, because a shorter makespan is likely to be desired by organisations.
Chapter 4

Decoupling Quantitative Task Networks

In the previous chapter, we studied plan decoupling as a plan-coordination mechanism that is acceptable for self-interested agents that have to complete a set of tasks with qualitative-temporal constraints. By decoupling such a qualitative task network, the agents are enabled to construct plans (i.e., add qualitative constraints) for their own parts of the task network independently from each other, while being guaranteed that their constructed plans are not in conflict with each other.

However, in many real-life situations, task-based planning problems need to be solved for completing a set of tasks that are also constrained by additional quantitative-temporal information, such as task durations and time windows. One problem in which such quantitative-temporal constraints are used is that of aircraft ground handling discussed in Section 1.1.1. There, aircraft need servicing agents to complete sets of ground-handling tasks (e.g., (de)boarding passengers and cabin cleaning) that are constrained by a precedence relation (e.g., deboarding before boarding). In addition, all ground-handling tasks for a certain aircraft have to be completed in a time window (e.g., between 05:30 and 07:20). Clearly, when such quantitative-temporal constraints are needed, the framework of the previous chapter does not suffice.

In order to complete the constrained set of tasks, the agents need to construct plans and schedules for it. As such, constructing plans and schedules can be regarded as adding qualitative- and quantitative-temporal constraints to the task network, respectively, similar to what we have seen in the previous chapter. For instance, consider a refuelling agent that has to complete two tasks $R(plane1), R(plane2)$ between 05:50 and 06:40, where each task requires 10 to 20 minutes to complete. The agent can construct a plan that adds the qualitative-temporal constraint $R(plane1)$ before $R(plane2)$. A schedule for it might be to execute task $R(plane1)$ between 05:55 and 06:10, and task $R(plane2)$ between 06:20 and 06:40 (i.e., the quantitative-temporal constraints to be satisfied). Like we have seen in the previous chapter, it must be guaranteed that the task network remains consistent after adding these qualitative- and quantitative-temporal constraints.
Here, two coordination problems arise, which we call the plan-coordination problem and the schedule-coordination problem. The plan-coordination problem arises when locally-consistent plans together might not allow a dispatching (e.g., when the precedence relation contains a cycle). The schedule-coordination problem arises when locally-consistent schedules might not be compatible with each other (i.e., there exists no dispatching that satisfies both). For instance, consider two tasks $\tau_1, \tau_2$ need to be completed between 05:50 and 06:40 by two agents $A_1, A_2$, where task $\tau_1$ is assigned to $A_1$ and task $\tau_2$ to $A_2$. When these tasks take 5 minutes each and need to be executed in the order $\tau_1$ before $\tau_2$, a conflict arises when $A_1$ constructs the schedule to execute task $\tau_1$ between 06:15 and 06:25 and agent $A_2$ schedules its task between 06:05 and 06:12.

For solving these coordination problems, we take an a-priori coordination-by-design approach. The idea is to achieve plan/schedule coordination by changing the original problem instance by adding a set of constraints, called a coordination set.

As we explained in Chapter 1, we propose a three-phase approach for completing a set of tasks. These phases are, in sequence, a (strategic) planning phase, a (tactical) scheduling phase, and an (operational) dispatching phase. In the planning phase, we enable agents to express their preferences on the execution order of their assigned tasks (i.e., they can add qualitative-temporal constraints). In the scheduling phase, we enable agents to add their preferences on the execution times of their assigned tasks (i.e., they can add quantitative-temporal constraints). Here, plan and schedule decoupling are used to coordinate at the beginning of the planning and scheduling phase, respectively. Finally, in the dispatching phase, the agents need to construct dispatchings that satisfy all expressed preferences (i.e., the qualitative- and quantitative-temporal constraints). To enable agents to independently construct dispatchings for their assigned tasks, temporal decoupling (Hunsberger, 2002) is available as an a-priori coordination-by-design approach (see Section 2.3.2).

The goal of this chapter is to study plan- and schedule-coordination problems for task networks with quantitative-temporal constraints. Here, we extend the applicability of plan decoupling to a more expressive framework (i.e., task networks with quantitative-temporal constraints), and extend the decoupling approach to allow agents to independently add more expressive constraints (i.e., quantitative-temporal constraints). More specifically, we are interested in comparing the computational complexity of plan decoupling in this quantitative-temporal framework (i) to schedule decoupling, and (ii) to plan decoupling in the qualitative-temporal framework of the previous chapter. Therefore, we require that the qualitative-temporal framework can be embedded in the quantitative-temporal framework used in this chapter. Finally, we want to determine the relation between the plan- and schedule-coordination problem.

This chapter is structured as follows. In Section 4.1, we define a framework for specifying sets of quantitatively-constrained tasks that are assigned to agents, which framework extends the qualitative task network of the previous chapter. In Section 4.2, we define and study plan decoupling as an a-priori coordination mechanism for independently adding qualitative-temporal constraints. In Section 4.3,
we study the schedule coordination that is needed when allowing agents to independently add quantitative-temporal constraints (subsuming qualitative-temporal constraints). In Section 4.4, we define dispatch coordination and study its relation to schedule coordination. In Section 4.5, we study the relation between plan and schedule coordination. We conclude this chapter by discussing the applicability of the described approach, and provide directions for future work in Section 4.6.

4.1 Quantitative Task Networks

In the previous chapter, we considered task networks consisting of qualitative-temporal constraints. In practice, however, a more expressive framework is needed that allows quantitative-temporal constraints to be used, such as

- **Time windows**
  Deadlines are often imposed before which tasks need to be completed, or intervals within which tasks need to be completed (e.g., between 09:00 and 17:00).

- **Task durations**
  Duration of a task can often be estimated (e.g., travelling time during rush hour), or is exactly known (e.g., boil the egg for 4 minutes).

- **Task separation times**
  Some tasks are not allowed to take place too close in time (e.g., minimum distance between ships loaded with inflammable chemicals).

Although other—more general—types of quantitative constraints can be thought of (e.g., disjunctions of time windows as used in temporal constraint satisfaction problems (Dechter et al., 1991)), we will focus on the listed types of constraints.

In this section, we will present a quantitative task network called the *simple temporal task network* (STTN), which extends the qualitative-task network formalism discussed in the previous chapter to allow time windows, task durations, and task separation times.

**Simple Temporal Task Networks**

We consider simple task networks \( T, C \) consisting of a set of elementary tasks \( T = \{ \tau_1, \tau_2, \ldots, \tau_k \} \) that are constrained by a set of temporal constraints \( C \). Each task \( \tau \in T \) is represented as an ordered pair of time-point variables \((t^s, t^e)\), with \( t^s \) being the start and \( t^e \) being the end of the execution of the task \( \tau \), and each set of tasks \( T \) is represented by its set of time-point variables \( T = \{ t^s, t^e \mid \tau \in T \} \).

In order to constrain the execution of a task to a time window, each time-point variable \( t \) is constrained to an interval \( I(t) = [lb(t), ub(t)] \), with \( lb(t) \leq ub(t) \), \( lb(t) \in \mathbb{Q} \cup \{-\infty\} \) and \( ub(t) \in \mathbb{Q} \cup \{\infty\} \). The most relaxed interval \((-\infty, \infty)\) is called the *universal time window*. 
In order to constrain task duration and task separation times, a function $C : T \times T \rightarrow \mathbb{Q} \cup \{\infty\}$ is defined that specifies an upper bound on the temporal distance for every ordered pair of time-point variables. For any ordered pair of time points $t_i, t_j$, the constraint $t_j - t_i \leq \delta_{ij}^+$ specifies an upper bound on the temporal distance between the time-point variables. A lower bound constraint $\delta_{ij}^\leq t_j - t_i$ on the temporal distance between any ordered pair of time points $t_i, t_j$ can be specified as an upper bound constraint on the reversed order $t_i - t_j \leq \delta_{ji}^+$. To improve readability, we write $t_j - t_i \in [\delta_{ij}^-, \delta_{ij}^+]$ for the constraint on the distance from $t_i$ to $t_j$.

**Example 4.1.** Consider two tasks $\tau_1, \tau_2$ that each require between 20 and 30 minutes. Furthermore, these tasks need to be executed in sequence $\tau_1$ before $\tau_2$ in the time window 06:00 and 07:15, where the tasks have a minimum separation time of 5 minutes (i.e., task $\tau_2$ must start at least 5 minutes after the completion of task $\tau_1$).

This situation can be represented as an STTN as follows. The durations of tasks $\tau_1 = (t_1^\tau, t_2^\tau)$ and task $\tau_2 = (t_1^\nu, t_2^\nu)$ are constrained by $t_1^\tau - t_1^\nu \in [20, 30]$ and $t_2^\nu - t_2^\tau \in [20, 30]$, respectively. We represent the minimum separation time by the constraint $t_2^\nu - t_1^\tau \in [5, \infty]$ (i.e., $t_1^\tau - t_2^\nu \leq -5$). The temporal distance of all other ordered pairs of time-point variables $(t_i, t_{ij})$ is constrained by $t_j - t_i \in [-\infty, \infty]$. Finally, we constrain the execution times of all time-point variables by $\forall t \in \{t_1^\tau, t_2^\nu, t_1^\nu, t_2^\tau\} : I(t) = [0, 75]$, representing 06:00 by the value 0.

Furthermore, we assume the set of tasks to be partitioned $T = \{T_i\}_{i=1}^n$ such that each partition $T_i$ is assigned to agent $A_i$. Now, each agent $A_i$ is responsible for completing its sub-task network $(T_i, C_i, I_i)$ induced by the task allocation, while it is the agents’ joint responsibility to meet the inter-agent constraints $C_{inter}$.

The STTN can thus be represented as the tuple $\langle\{\langle T_i, C_i, I_i \rangle\}_{i=1}^n, C_{inter}\rangle$.

Similar to Definition 3.6, we define a solution to an STTN, which we refer to as a dispatching, as a value assignment $\nu$ for all time-point variables that satisfies all constraints.

**Definition 4.2 (Dispatching).** A dispatching $\nu$ for an STTN $S = \langle T, C, I \rangle$ is a function $\nu : T \rightarrow \mathbb{Q}$ such that:

1. $\forall t \in T : \nu(t) \in I(t)$, and

2. $\forall t_i, t_j \in T : c_{ij} : t_j - t_i \leq \delta_{ij} \in C$ implies $\nu(t_j) - \nu(t_i) \leq \delta_{ij}$.

An STTN is called consistent, when there exists at least one dispatching $\nu$ for it. In order to check the consistency of an STTN, we can make use of the relation with the STN (Dechter et al., 1991) formalism.\footnote{It is well known (Dechter et al., 1991) that an STN is consistent if, and only if, its associated distance graph does not have any negative cycles. Moreover, there exist polynomial-time procedures for deciding the consistency and finding a dispatching $\nu$ for an STN (Dechter et al., 1991).}

An unpartitioned STTN $\langle T, C, I \rangle$ can be translated into an STN\footnote{Despite the fact that STTNS are subsumed by STNs, we prefer using STTN notation due to the similarity with the notation used in the framework for qualitative-temporal task networks.} $\langle T', C' \rangle$ by adding a fixed-reference point $z$ to the set of time-point variables (i.e., $T' = T \cup \{z\}$),
and translating each time window $I(t) = [lb(t), ub(t)]$ to constraints between $z$ and $t$ (i.e., $z - t \leq -lb(t)$ and $t - z \leq ub(t)$). Hence, the available techniques and results from the STN literature can be used for STTNs.

An STTN $\langle \{T_i\}_{i=1}^n, C, I \rangle$ can be represented as a partitioned labelled directed graph $G = (\{T_i\}_{i=1}^n, A)$. Here, each vertex $t_i$ is a direct representative of the corresponding time-point variable, and a (directed) arc $(t_i, t_j)$ is labelled $[\delta_{ij}^-, \delta_{ij}^+]$ to represent the constraint $t_j - t_i \in [\delta_{ij}^-, \delta_{ij}^+]$. Furthermore, each vertex $t_i$ is labelled with the time window $I(t_i)$ of the time-point variable it represents.

A similar representation can be obtained by first translating the STN to an STN, which can then be represented as a labelled directed graph, which is called the distance graph (Dechter, 2003). In this representation, each vertex represents a time-point variable, and a (directed) arc $(t_i, t_j)$ is labelled $\delta_{ij}^+$ for the constraint $t_j - t_i \leq \delta_{ij}^+$. Although this distance-graph representation is commonly used in the STN-related literature, we prefer the former interval notation for STTNs because it requires fewer arcs.

In the remainder of this thesis, we assume that all constraints in the STTN are the strongest implied constraints. In the STN literature, such a network is called a minimal network in which each pair of time-point variables is constrained by the strongest implied constraint (Dechter, 2003). It is well known that such a minimal network can be obtained by applying an all-pair shortest-path (APSP) algorithm to the distance graph (Cormen et al., 1990; Dechter et al., 1991). For more information, we refer the reader to the cited literature. In the remainder, we assume the constraints in each STTN to be the strongest implied constraints, which can be obtained by translating the STTN into an STN, apply an APSP algorithm, and updating the STTN based on the STN’s minimal network.

### 4.1.1 Plans in STTNs

Agents need to construct (temporal) plans for executing their assigned sets of tasks. Planning is the act of imposing ordering constraints (i.e., qualitative constraints) on tasks compatible with the existing constraints. Therefore, in the previous chapter, we defined plans for qualitative task networks as extensions of the existing precedence and synchronisation relations. This idea will also be used to define plans in quantitative-temporal networks.

First, we show how a qualitative task network of the previous chapter is translated into an STTN. Thereafter, we define a plan for an STTN and show that this extends the definition of a plan for a qualitative task network.

Given a qualitative task network $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$, we can construct an equivalent STTN $\langle \{T_i\}_{i=1}^n, C, I \rangle$. Here, we assume an $\epsilon > 0$ to be available that represents the smallest temporal difference possible between two time-point variables in the system used.

1. The partitioned set of time-point variables $\{T_i\}_{i=1}^n$ coincides with that of the qualitative task network.
2. To encode the ordering relations $\prec$, $\equiv$, we define the temporal-distance function $\mathcal{C}$ for every ordered pair $(t_i, t_j)$ as follows:

(a) $c_{ij} : t_j - t_i \leq -\epsilon$ if $t_j < t_i$,
(b) $c_{ij} : t_j - t_i \leq 0$ if $t_j \equiv t_i$,
(c) $c_{ij} : t_j - t_i \leq \infty$ otherwise.

3. Because no earliest starting time or deadline restrictions are imposed on the time-point variables, we only have universal time windows: $\forall t \in T_i : I(t) = (-\infty, \infty)$.

Because planning involves adding ordering and synchronisation constraints, we need to define what it means to add such a constraint to an STTN. For this purpose, we consider a single temporal-distance constraint $c_{ij}$ for an ordered pair $(t_i, t_j)$. Recall that, in an STTN, there exists a temporal-distance constraint $c_{ij} : t_j - t_i \leq \delta_{ij}$ for every ordered pair of time-point variables $(t_i, t_j)$. In an STTN, adding a qualitative-temporal constraint between time-point variables $t_i, t_j$ is achieved by tightening the constraint $c_{ij}$, called a qualitative refinement, as follows.

Imposing a qualitative-temporal constraint $t_j \prec t_i$ or $t_j \equiv t_i$ corresponds to the following qualitative refinements.

1. $t_j \prec t_i$ corresponds to $c_{ij} : t_j - t_i \leq -\epsilon$, and
2. $t_j \equiv t_i$ corresponds to $c_{ij} : t_j - t_i \leq 0$ and $c_{ji} : t_i - t_j \leq 0$.

**Definition 4.3 (Qualitative refinement).** A qualitative refinement of a constraint $c_{ij} : t_j - t_i \leq \delta_{ij} \in \mathcal{C}$ in an STTN $\mathcal{S} = \langle T, \mathcal{C}, I \rangle$ is a tightening $c'_{ij} : t_j - t_i \leq \delta'_{ij}$, with $\delta'_{ij} \in \{-\epsilon, 0\}$ and $\delta'_{ij} < \delta_{ij}$.

A qualitative refinement of an STTN $\mathcal{S} = \langle T, \mathcal{C}, I \rangle$ is an STTN $\mathcal{S}' = \langle T, \mathcal{C}', I \rangle$ such that each constraint $c'_{ij} \in \mathcal{C}'$ is either equal to the original constraint $c_{ij} \in \mathcal{C}$, or is a qualitative refinement thereof. Note that making such a qualitative refinement is similar to adding qualitative constraints in a qualitative task network of the previous chapter (i.e., constructing a plan). Therefore, we define a plan $\mathcal{C}^\pi$ for an STTN $\mathcal{S}$ as follows.

**Definition 4.4 (Plan).** A plan $\mathcal{C}^\pi$ for an STTN $\mathcal{S} = \langle T, \mathcal{C}, I \rangle$ is a set of constraints for which it holds that each constraint $c'_{ij} \in \mathcal{C}^\pi$ is either equal to the original constraint $c_{ij} \in \mathcal{C}$, or $c'_{ij}$ is a qualitative refinement of the constraint $c_{ij} \in \mathcal{C}$.

A plan $\mathcal{C}^\pi$ for an STTN $\mathcal{S} = \langle T, \mathcal{C}, I \rangle$ is consistent, when there exists a dispatching for the STTN $\mathcal{S}^\pi = \langle T, \mathcal{C}^\pi, I \rangle$. When such a dispatching does not exist, the plan is inconsistent.

**Remark 4.5.** Of course, it is not always possible to qualitatively refine a constraint. While qualitative-temporal constraints can be added only when no constraint exists between two time-point variables in a qualitative task network, the qualitative-temporal constraints that can be added to an STTN depend on the values of $\delta_{ij}$. For
instance, the constraint \( t_j - t_i \in [-50, 50] \) allows \( t_i < t_j \), \( t_j < t_i \), and \( t_j \equiv t_i \), while the constraint \( t_j - t_i \in [0, 50] \) allows only \( t_i < t_j \) and \( t_j \equiv t_i \).

### 4.1.2 Plan Coordination

In the previous section, we have described what plans can be constructed for completing a quantitative task network. When multiple agents are involved, the tasks are partitioned and assigned to agents such that each agent is capable of completing its set of tasks. However, like in planning for qualitative task networks, this local consistency does not guarantee the consistency of these plans together.

![Example STTN](image)

(a) Example STTN

![Planning conflict](image)

(b) Planning conflict

Figure 4.1: Independent planning can result in a conflict.

**Example 4.6.** Consider the STTN depicted in Figure 4.1(a). There are two agents \( A_1 \) and \( A_2 \), each having to complete two quantitatively-constrained tasks. Suppose that each agent decides to construct plans that correspond with adding the precedence constraints as indicated by dashed arrows in Figure 4.1(b), and each agent normalises its set of constraints accordingly. Since the precedence constraints are cyclic, it is not difficult to see that there cannot exist a feasible joint dispatching for this task network.

Note that the conflict given in the example above is closely related to the inter-agent directed cycles that needed to be prevented in the previous chapter. A coordination problem must be solved for ensuring that for each possible *joint plan* \( C^* = \{C^*_i\}^n_{i=1} \), consisting of locally-consistent plans, there exists a *joint dispatching* that also satisfies all constraints in \( C_{\text{inter}} \).

**Definition 4.7 (Joint Dispatching).** A joint dispatching \( \nu = \{\nu_i\}^n_{i=1} \) for an STTN \( \langle \{T_i, C_i, I_i\}^n_{i=1}, C_{\text{inter}} \rangle \) is a function \( \nu : T \rightarrow \mathbb{Q} \) such that:

1. \( \nu_i \) is a dispatching for \( \langle T_i, C_i, I_i \rangle \),
2. \( \forall t : t \in T_i \) implies \( \nu(t) = \nu_i(t) \), and
3. \( \forall t_i \in T_i, t_j \in T_{j \neq i} : c_{ij} : t_j - t_i \leq \delta_{ij} \in C_{\text{inter}} \) implies \( \nu(t_j) - \nu(t_i) \leq \delta_{ij} \).
Similar to joint plan consistency in qualitative task networks, we define a joint plan to be consistent when there exists a joint dispatching for it (cf. Definition 3.10).

**Definition 4.8 (Joint Plan Consistency).** A joint plan \( \{C^\pi_i\}_{i=1}^n \) for STTN \( \langle\{(T_i, C_i, I_i)\}_{i=1}^n, C_{inter}\rangle \) is called consistent if there exists a joint dispatching \( \nu \) for \( \langle\{(T_i, C^\pi_i, I_i)\}_{i=1}^n, C_{inter}\rangle \).

Similar to plan coordination in qualitative task networks, we call an STTN \( S \) to be plan coordinated when the joint plan consistency is guaranteed for every combination of locally-consistent plans (cf. Definition 3.11).

**Definition 4.9 (Plan Coordination).** An STTN \( S = \langle\{(T_i, C_i, I_i)\}_{i=1}^n, C_{inter}\rangle \) is called plan coordinated when the joint plan \( C^\pi = \{C^\pi_i\}_{i=1}^n \) is consistent, for every set \( \{C^\pi_i\}_{i=1}^n \) of locally-consistent plans \( C^\pi_i \).

### 4.2 Plan Decoupling in STTNs

In the previous section, a framework was presented for describing quantitative task networks (i.e., STTNs) that are assigned to a set of agents. We showed that a plan-coordination problem may arise when these agents construct plans for their sets of tasks, even when all plans are locally consistent. We solve this plan-coordination problem by constructing a plan decoupling of the task network.

**Definition 4.10 (Plan Decoupling).** A plan decoupling for an STTN \( \langle\{(T_i, C_i, I_i)\}_{i=1}^n, C_{inter}\rangle \) is an STTN \( \langle\{(T_i, C'_i, I_i)\}_{i=1}^n, C_{inter}\rangle \), such that

1. each \( (T_i, C'_i, I_i) \) is a qualitative-refined sub-task network of \( (T_i, C_i, I_i) \), and
2. the STTN \( \langle\{(T_i, C'_i, I_i)\}_{i=1}^n, C_{inter}\rangle \) is plan coordinated.

Like plan decoupling of qualitative task networks, we intend to achieve a plan decoupling of an STTN by adding a coordination set \( \Gamma = \bigcup_{t=1}^n \Gamma_t^\prec \cup \bigcup_{t=1}^n \Gamma_t^\equiv \), with \( \Gamma_t^\prec \subseteq T_t \times T_t \) and \( \Gamma_t^\equiv \subseteq T_t \times T_t \). For each sub-task network \( (T_i, C_i, I_i) \), adding the set of constraints \( \Gamma_i = \Gamma_i^\prec \cup \Gamma_i^\equiv \) results in the qualitatively-refined sub-task network \( \langle T_i, C'_i, I_i \rangle \), where each constraint in \( \Gamma_t \) is used in a qualitative refinement in \( C'_i \).

Such a coordination set is guaranteed to exist for every consistent STTN.

**Proposition 4.11.** For every consistent STTN \( S = \langle\{(T_i, C_i, I_i)\}_{i=1}^n, C_{inter}\rangle \), there exists a coordination set \( \Gamma = \bigcup_{t=1}^n \Gamma_t^\prec \cup \bigcup_{t=1}^n \Gamma_t^\equiv \) such that \( \langle\{(T_i, C'_i, I_i)\}_{i=1}^n, C_{inter}\rangle \) forms a plan decoupling.

**Proof.** Consider an STTN \( S = \langle\{(T_i, C_i, I_i)\}_{i=1}^n, C_{inter}\rangle \). Since \( S \) is consistent, there exists at least one dispatching \( \nu \) for it. Let us define the sets

- \( \prec^* = \{ t - t' \leq -\epsilon \mid \nu(t) < \nu(t') \} \), and
- \( \equiv^* = \{ t - t' \leq 0 \mid \nu(t) = \nu(t') \} \).
4.2. PLAN DECOUPLING IN STTNs

Here, $\prec^* \cup \equiv^*$ defines a total ordering on the set of time-point variables $\{T_i\}_{i=1}^n$ (i.e., between each pair of time-point variables there exists a constraint in either $\prec^*$ or $\equiv^*$).

Consider the qualitatively-refined sub-task network $S'_i = \langle T_i, C'_i, I_i \rangle$ of sub-task network $S_i$, where the following sets of qualitative constraints are used in the qualitative refinements in $C'_i$.

- $\prec^*_i = (\prec^* \cap (T_i \times T_i))$, and
- $\equiv^*_i = (\equiv^* \cap (T_i \times T_i))$.

Because $\prec^*_i \cup \equiv^*_i$ defines a total ordering on the set of time-point variables $T_i$, $C'_i$ defines a total ordering on $T_i$. Hence, there are no further qualitative refinements in any sub-task network possible, such that agent $A_i$ constructs the plan $C_i^\pi = C'_i$.

Clearly, the joint plan $C^\pi = \{C'_i\}_{i=1}^n$ is consistent, since it allows the dispatching $\nu$.

The coordination set can now be constructed as $\Gamma = \bigcup_{i=1}^n \Gamma^\prec_i \cup \bigcup_{i=1}^n \Gamma^\equiv_i$, with $\Gamma^\prec_i = \prec^*_i$ and $\Gamma^\equiv_i = \equiv^*_i$.

Now that we are guaranteed that there always exists a coordination set that plan decouples an STTN, we can define the STTN PLAN-DECOUPLING PROBLEM (STTN-PDP) which is the problem of finding such a set.

**STTN Plan-Decoupling Problem (STTN-PDP)**

Find a coordination set $\Gamma = \bigcup_{i=1}^n \Gamma^\prec_i \cup \bigcup_{i=1}^n \Gamma^\equiv_i$ for an STTN $\langle\{\langle T_i, C_i, I_i \rangle\}_{i=1}^n, C_{inter} \rangle$, such that $\langle\{\langle T_i, C_i^\Gamma, I_i \rangle\}_{i=1}^n, C_{inter} \rangle$ is a plan decoupling.

**Corollary 4.12.** STTN-PDP can be solved in polynomial time.

*Proof.* For any consistent STTN $S$, a dispatching $\nu$ can be found in polynomial time by translating it to an equivalent STN for which a solution can be found in polynomial time. Then, a coordination set can be found by following the construction used in the proof of Proposition 4.11. Because this construction is done in polynomial time, the corollary follows.

Now that we know that an arbitrary coordination set can be found in polynomial time, we continue with studying the complexity of the variant where the coordination set must consist of a minimum number of constraints.

We will define and study a recognition and a construction variant for plan decoupling in STTNs. These problems for STTN are related to their counterparts for qualitative task networks, which we defined and studied in Section 3.2.1. The first variant is the recognition variant of plan decoupling in STTNs, called STTN PLAN-DECOUPLING RECOGNITION PROBLEM (STTN-PDRP), which is defined formally as follows.
CHAPTER 4. DECOUPLING QUANTITATIVE TASK NETWORKS

STTN Plan-Decoupling Recognition Problem (STTN-PDRP)

INSTANCE: An STTN \( \langle \{ (T_i, C_i, I_i) \}^{n}_{i=1}, C_{\text{inter}} \rangle \) and a set of constraints \( \Gamma = \bigcup_{i=1}^{n} \Gamma_i^{\prec} \cup \bigcup_{i=1}^{n} \Gamma_i^{\equiv} \).

QUESTION: Does it hold that the qualitatively-refined STTN \( \langle \{ (T_i, C_i^\Gamma_i, I_i) \}^{n}_{i=1}, C_{\text{inter}} \rangle \) is plan coordinated?

Proposition 4.13. STTN-PDRP is coNP-complete.

Proof. Membership is due to the fact that a no-certificate can be verified in polynomial time. Such a no-certificate consists of a set of qualitative refinements that can be added to the STTN \( S' = \langle \{ (T_i, C_i^\Gamma_i, I_i) \}^{n}_{i=1}, C_{\text{inter}} \rangle \) in polynomial time such that each qualitatively-refined sub-task network \( S_i^\Gamma \) of agent \( A_i \) remains consistent while the qualitatively-refined STTN \( S'' \) of STTN \( S' \) has become inconsistent. Checking the consistency of each refined sub-task network \( S_i^\Gamma \) as well as the inconsistency of the STTN \( S'' \) can be done in polynomial time (Dechter, 2003).

Hardness is due to qualitative task networks (discussed in Chapter 3) being subsumed by STTNs, and that the plan-recognition problem for the former has been proven to be coNP-complete (see Proposition 3.22).

In general, a plan-decoupling problem remains to be solved for an STTN \( S \). Here, we want to achieve a plan decoupling by finding a coordination set \( \Gamma = \bigcup_{i=1}^{n} \Gamma_i^{\prec} \cup \bigcup_{i=1}^{n} \Gamma_i^{\equiv} \) with a minimum number of constraints. The constraints in this coordination set are used to determine the qualitatively-refined task network \( S' = \langle \{ (T_i, C_i^\Gamma_i, I_i) \}^{n}_{i=1}, C_{\text{inter}} \rangle \) which is plan coordinated. The problem of finding such a smallest coordination set is called the STTN CARDINAL-MINIMAL PLAN-DECOUPLING PROBLEM (STTN-cMPDP), for which we define the decision variant as follows.

Decision variant of the STTN-cMPDP

INSTANCE: An STTN \( \langle \{ (T_i, C_i, I_i) \}^{n}_{i=1}, C_{\text{inter}} \rangle \) and a positive integer \( K \).

QUESTION: Does there exist a coordination set \( \Gamma = \bigcup_{i=1}^{n} \Gamma_i^{\prec} \cup \bigcup_{i=1}^{n} \Gamma_i^{\equiv} \) with \( |\Gamma| \leq K \) such that the qualitatively-refined STTN \( \langle \{ (T_i, C_i^\Gamma_i, I_i) \}^{n}_{i=1}, C_{\text{inter}} \rangle \) is plan coordinated?

Proposition 4.14. STTN-cMPDP is \( \Sigma_2^p \)-complete.

Proof. Membership is due to the fact that a yes-certificate can be verified in non deterministic polynomial time: Given an instance \( S = \langle \{ (T_i, C_i, I_i) \}^{n}_{i=1}, C_{\text{inter}} \rangle \). Non-deterministically choose a coordination set \( \Gamma \) of at most \( K \) qualitative refinements and check (in polynomial time) that the STTN \( S' \) resulting from qualitatively refining \( C \) with this coordination set is consistent. Now, verify in polynomial time whether \( S' = \langle \{ (T_i, C_i^\Gamma_i, I_i) \}^{n}_{i=1}, C_{\text{inter}} \rangle \) is an STTN-PDRP yes-instance using an STTN-PDRP-oracle. Since such an oracle is a coNP-oracle, the problem is in \( \text{NP}^{\text{coNP}} = \Sigma_2^p \).
Hardness is due to qualitative task networks being subsumed by STTNs, hence, the cardinal-minimal plan-decoupling problem for qualitative task networks is a special case of the STTN-cMPDP. Therefore, the problem is $\Sigma_2^p$-complete (see Proposition 3.25).

In this section, we extended the applicability of plan decoupling to STTNs. We have shown that, despite the added expressiveness of the task network, the plan-decoupling problems do not increase in terms of computational complexity in comparison with their counterparts for qualitative task networks.

### 4.3 Schedule Coordination

In the previous section, we studied plan decoupling in STTNs. Plan decoupling an STTN allows the agents to independently construct locally-consistent plans, while being guaranteed that their joint plan is consistent. Besides independent planning, self-interested agents want to construct schedules for completing their sets of tasks. Similar to the plan-coordination problem, we need to guarantee the consistency of every combination of locally-consistent schedules. In this section, we define the schedule-coordination problem that needs to be solved when allowing agents to independently construct their local schedules.

First, let us define a schedule in an STTN. Recall that planning allows agents to qualitatively refine the constraints on their tasks (i.e., their local STTNs). For constructing schedules, we allow agents to quantitatively refine the constraints on their tasks. We define these quantitative refinements as follows.

**Definition 4.15 (Quantitative refinement).** A quantitative refinement of a constraint in an STTN $S = \langle T, C, I \rangle$ is

- a tightening $c'_{ij} : t_j - t_i \leq \delta'_{ij}$ of a constraint $c_{ij} \in C$, with $\delta'_{ij} < \delta_{ij}$, or
- a tightening $I'(t) = [lb'(t), ub'(t)]$ of a time window $I(t) = [lb(t), ub(t)]$, with $lb(t) < lb'(t)$ or $ub'(t) < ub(t)$.

A quantitative refinement of an STTN $S = \langle T, C, I \rangle$ is an STTN $S' = \langle T, C', I \rangle$ such that each constraint $c'_{ij} \in C'$ is either equal to the original constraint $c_{ij} \in C$, or is a quantitative refinement thereof. We associate these quantitative refinements with the construction of schedules, and, therefore, define a schedule $C^\sigma$ for an STTN $S$ as follows.

**Definition 4.16 (Schedule).** A schedule $C^\sigma$ for an STTN $S = \langle T, C, I \rangle$ is a set of constraints for which it holds that each constraint $c'_{ij} \in C^\sigma$ is either equal to the original constraint $c_{ij} \in C$, or $c'_{ij}$ is a quantitative refinement of the constraint $c_{ij} \in C$. 

CHAPTER 4. DECOUPLING QUANTITATIVE TASK NETWORKS

Given an STTN $S = \langle T, C, I \rangle$ and a schedule $C^\sigma$ for it, adding the scheduling constraints $C^\sigma$ to the STTN $S$ results in the updated STTN $C^\sigma = \langle T, C^\sigma, I \rangle$. Clearly, it holds that a dispatching for the updated STTN $C^\sigma$ also is a dispatching for the STTN $S$.

Now, we call a schedule $C^\sigma$ for an STTN $S = \langle T, C, I \rangle$ to be consistent, when there exists a dispatching for the STTN $C^\sigma = \langle T, C^\sigma, I \rangle$. When such a dispatching does not exist, the schedule is inconsistent.

**Remark 4.17.** Note that we differentiate between a schedule and a dispatching. While a dispatching is a fixed assignment of values to the time-point variables, a schedule allows multiple dispatchings. Furthermore, these schedules don’t need to specify absolute moments in time, but constrain the execution of tasks relative to each other.

**Example 4.18.** Consider the container transshipment for a large container ship. The agent that is responsible for the transshipment is typically rewarded based on the amount of time between mooring and the completion of the transshipment. In order to receive the highest reward, a plan is constructed that allows the transshipment to be completed in the shortest amount of time, measured from start to finish. To guarantee that this plan will also be executed in the shortest amount of time, the agent constructs a schedule that achieves this (by tightening the constraints as little as possible). This schedule was constructed relative to the mooring of the ship. When the ship is finally moored, a dispatching for the constructed schedule can be determined which all result in the highest reward for the agent.

Next, consider the situation for scheduling by a set of agents, that construct locally-consistent schedules. Because these agents want to construct their schedules independently from the other agents, a coordination problem may arise. Although all constructed schedules are locally consistent, the consistency of these schedules together is not guaranteed on beforehand. Clearly, a coordination mechanism is required to guarantee the consistency of such locally-consistent schedules together, as illustrated by the following example.

**Example 4.19.** Consider the example in Figure 4.2(a), together with the following value assignments as quantitative refinements. Agent $A_1$ might come up with $\nu_1(t^*_1) = 20$, $\nu_1(t^*_2) = 30$, $\nu_1(t^*_3) = 10$, and $\nu_1(t^*_4) = 20$. Agent $A_2$, on the other hand, has decided upon $\nu_2(t^*_1) = 23$, $\nu_2(t^*_2) = 24$, $\nu_2(t^*_3) = 12$, and $\nu_2(t^*_4) = 16$. Both quantitative refinements are dispatchings because they satisfy the local constraints on the time-points of their sub-task networks. However, they do not constitute a joint dispatching for the complete task as they obviously violate the inter-agent constraint $t^*_4 - t^*_3 \in [0, 0]$ (i.e., $\nu_1(t^*_4) - \nu_2(t^*_3) = 10 - 16 = -6 \notin [0, 0]$).

Because each agent $A_i$ constructs a local schedule $C^\sigma_i$ that is consistent with its intra-agent constraints $C_i$, the inconsistency of the joint schedule $\{C^\sigma_i\}_{i=1}^n$ has to be due to the violation of (a combination of) inter-agent constraints. Similar to plan
4.4. DISPATCH COORDINATION

4.4.1 Schedule Coordination

In the previous section, we discussed schedule coordination. When an STTN is schedule coordinated, the agents are enabled to quantitatively refine their local task networks without any restrictions on those refinements (i.e., the set of quantitative refinements only needs to be locally consistent). A variant of this schedule-coordination problem is to restrict the quantitative refinements that the agents are allowed to make to the tightest quantitative refinements only (i.e., dispatchings). In this section, we define and study the variant, called dispatch coordination, in which each agent must refine all constraints in its local task network with such tightest refinements, and study its relation to schedule coordination of the previous section.

Definition 4.20 (Schedule Coordination). An STTN $S = \langle\{\langle T_i, C_i, I_i \rangle\}_{i=1}^{n}, C_{\text{inter}}\rangle$ is called schedule coordinated when, for every set of locally-consistent schedules $\{C_i^\sigma\}_{i=1}^{n}$, there exists a joint dispatching $\{\nu_i\}_{i=1}^{n}$ for $S^\sigma = \langle\{\langle T_i, C_i^\sigma, I_i \rangle\}_{i=1}^{n}, C_{\text{inter}}\rangle$.

4.4 Dispatch Coordination

In dispatch coordination, it must be guaranteed that the merge of all local dispatchings together must be such a joint dispatching. Note that this guarantee is much stronger than only the existence of a joint dispatching (cf. Definition 4.20).

Definition 4.21 (Dispatch Coordination). An STTN $S = \langle\{\langle T_i, C_i, I_i \rangle\}_{i=1}^{n}, C_{\text{inter}}\rangle$ is called dispatch coordinated when every set of local dispatchings $\{\nu_i\}_{i=1}^{n}$ is a joint dispatching for $S$.

Proposition 4.22. Dispatch coordination can be achieved by temporal decoupling.

Proof. In temporal decoupling Hunsberger (2002), it is guaranteed that merging any set of local solutions results in a joint solution. There, a solution is a value coordination (see Definition 4.9), schedule coordination is achieved when the inter-agent constraints are satisfied for all combinations of locally-consistent refinements. This requires that the existence of a joint dispatching $\{\nu_i\}_{i=1}^{n}$ for the quantitatively-refined STTN is guaranteed.

Definition 4.20 (Schedule Coordination). An STTN $S = \langle\{\langle T_i, C_i, I_i \rangle\}_{i=1}^{n}, C_{\text{inter}}\rangle$ is called schedule coordinated when, for every set of locally-consistent schedules $\{C_i^\sigma\}_{i=1}^{n}$, there exists a joint dispatching $\{\nu_i\}_{i=1}^{n}$ for $S^\sigma = \langle\{\langle T_i, C_i^\sigma, I_i \rangle\}_{i=1}^{n}, C_{\text{inter}}\rangle$.

4.4 Dispatch Coordination

In the previous section, we discussed schedule coordination. When an STTN is schedule coordinated, the agents are enabled to quantitatively refine their local task networks without any restrictions on those refinements (i.e., the set of quantitative refinements only needs to be locally consistent). A variant of this schedule-coordination problem is to restrict the quantitative refinements that the agents are allowed to make to the tightest quantitative refinements only (i.e., dispatchings). In this section, we define and study the variant, called dispatch coordination, in which each agent must refine all constraints in its local task network with such tightest refinements, and study its relation to schedule coordination of the previous section.

Definition 4.21 (Dispatch Coordination). An STTN $S = \langle\{\langle T_i, C_i, I_i \rangle\}_{i=1}^{n}, C_{\text{inter}}\rangle$ is called dispatch coordinated when every set of local dispatchings $\{\nu_i\}_{i=1}^{n}$ is a joint dispatching for $S$.

Proposition 4.22. Dispatch coordination can be achieved by temporal decoupling.

Proof. In temporal decoupling Hunsberger (2002), it is guaranteed that merging any set of local solutions results in a joint solution. There, a solution is a value
assignment to all time-point variables, which we call a dispatching. Because an STTN can also be translated into an STN, the proposition follows.

In a dispatching $\nu$ for an STTN $S$, each time-point variable $t$ is assigned a single value $\nu(t)$. Now, there exists a quantitatively-refined STTN $S'$ of STTN $S$ for which there exists only one dispatching, which is the dispatching $\nu$. In this STTN $S'$, all time-window constraints are $[lb(t), ub(t)]$ with $lb(t) = ub(t) = \nu(t)$, and the temporal-distance constraints are determined accordingly. As such, the STTN $S'$ consists of tightest quantitative refinements of the constraints in STTN $S$. The above shows that a quantitatively-refined STTN $S'$ can be constructed for an STTN $S$ such that dispatching $\nu$ is the only dispatching, based on any dispatching $\nu$ for STTN $S$.

When STTN $S$ is schedule coordinated, it allows any set of locally-consistent schedules to be constructed while guaranteeing that there exists a joint dispatching for the joint schedule. Because a dispatching is a locally-consistent schedule, every set of local dispatchings is guaranteed to have a joint dispatching. In fact, this joint dispatching is the merge of all these local dispatchings together. Hence, schedule coordination implies dispatch coordination.

Now, consider the situation in which STTN $S$ is dispatch coordinated. Dispatch coordination allows any set of local dispatchings to be constructed, that together form a joint dispatching. A local schedule reduces the set of local dispatchings such that at least one local dispatching remains. All sets of local dispatchings that also satisfy the local schedules form a joint dispatching for STTN $S$, because $S$ is dispatch coordinated. Hence, dispatch coordination implies schedule coordination, such that following proposition results.

**Proposition 4.23.** An STTN is schedule coordinated if, and only if, it is dispatch coordinated.

Because STTNs can be translated into an STN (see Section 4.1), temporal decoupling can be used for dispatch coordinating an STTN. Due to Proposition 4.23, we get the following corollary.

**Corollary 4.24.** A temporal-decoupling mechanism can be used for solving the schedule-coordination problem for STTNs.

**Example 4.25.** Consider the STTN in Figure 4.3(a), for which we will construct a temporal decoupling. As an example, consider the constraint $t_2 - t_1 \in [1, 14]$, where it must hold that $t_1 \in [20, 30]$ and $t_2 \in [21, 34]$. Clearly, it always holds that $t_2 - t_1 \leq 34 - 20 = 14 \in [1, 14]$. However, the lower bound given in the inter-agent constraint is not implied: $t_2 - t_1 \geq 21 - 30 = -9 \notin [1, 14]$. To guarantee this inter-agent constraint to hold, we must tighten the time windows of these time-point variables. In Figure 4.3(b), such a possible tightening is given in which the time windows are changed to $I(t_1) = [20, 25]$ and $I(t_2) = [26, 34]$. 

4.5 Relation Plan and Schedule Coordination

In the previous section, we showed that schedule coordination can be achieved with a temporal-decoupling mechanism. In this section, we will consider the relation between plan and schedule coordination.

In the next proposition, we show that plan coordination is implied when an STTN is schedule coordinated.

**Proposition 4.26.** If an STTN is schedule coordinated, then it is also plan coordinated.

**Proof.** When an STTN is schedule coordinated it guarantees that all sets of locally-consistent quantitative refinements together allow a joint dispatching. A qualitative refinement (see Definition 4.3) is a special type of quantitative refinement (see Definition 4.15). Because plan coordination requires the existence of a joint dispatching for a set of locally-consistent qualitative refinements, the proposition follows.

Therefore, the question arises if the plan-coordination problem should be solved separately at all. Can we not just apply a temporal-decoupling mechanism to solve all these coordination problems at once?

Recall from Chapter 1, that the agents need to make strategic and tactical decisions on the execution of their assigned sets of tasks. These levels of decision making require them to solve planning and scheduling problems, respectively. Based on the constructed plans and schedules, the agents want to impose preferences on the execution of their sets of tasks. The tactical decisions are reflected by the preferences on the time intervals in which the tasks are to be executed. The strategic decisions are expressed by the agents’ preferences on the execution order that stem from the constructed plans.

While plan coordination can be achieved by solving the schedule-coordination problem, this results in overconstrained solutions to the plan-coordination problem. In the next two examples, we illustrate such overconstrained schedule-coordination solutions. First, we consider an STTN in which sequential execution becomes impossible due to the schedule-coordination solution.
Example 4.27. Reconsider the example given in Figure 4.3(a), and suppose that agent $A_1$ prefers executing tasks $\tau_1$ and $\tau_4$ sequentially (i.e., either $\tau_1$ before $\tau_4$ or $\tau_4$ before $\tau_1$). With this in mind, consider the schedule-coordination solution given in Figure 4.3(b). In this solution, agent $A_1$ is unable to execute its tasks $\tau_1, \tau_4$ sequentially, because neither execution order allows a dispatching. However, in the initial situation both execution orders were allowed.

In the following example, the agents need to complete set of tasks that are constrained by qualitative-temporal constraints only. In these STTNS, all time windows are universal time windows. A schedule-coordination mechanism could arbitrarily choose time-window constraints that require each agent to execute all of its tasks in parallel (i.e., no ordering constraints among the tasks), as illustrated in the following example.

Example 4.28. Consider a supply chain of $n$ agents, where each agent $A_i$ needs to complete its task $\tau_{f_i}$ which is part of a sequence $s_f = \tau_{f_1} \text{ before } \tau_{f_2} \ldots \tau_{f_n}$. Assume that the task network consists of $m$ sequences $s_1, s_2, \ldots s_m$ of tasks that need to be completed.

A schedule-coordination solution might be to have $n$ non-overlapping time windows, such that each agent has to complete all its tasks within its time window before the next agent can start. Clearly, this is not a situation preferred in a supply chain.

The examples clearly show that applying a temporal-decoupling mechanism to achieve schedule coordination can result in an overconstrained solution to the plan-coordination problem. Therefore, we propose a two-phase approach that separates making the planning decisions (i.e., planning phase) from making the scheduling decisions (i.e., the scheduling phase). In each phase, we solve the plan and schedule-coordination problems a-priori in the respective phases. Then, each agent is allowed to add qualitative-temporal constraints associated with its planning preferences (or strategic decisions) in the planning phase, and to add quantitative-temporal constraints which are associated with its scheduling preferences (or tactical decisions) in the scheduling phase. The plan and schedule-coordination solutions guarantee that the added preferences will be satisfied without being revised—under the restriction of local consistency. As such, the a-priori plan and schedule-coordination mechanisms have become preference preserving.

4.6 Discussion

In this chapter, we studied plan- and schedule decoupling for a task network with quantitative-temporal constraints, called simple temporal task network (STTN). In these STTNS, quantitative-temporal constraints can be specified (e.g., time windows and task durations), thereby extending the qualitative task networks of Chapter 3. In this chapter, we presented the following three main contributions.
First, we studied the computational complexity of plan decoupling which is an a-priori coordination mechanism that enables agents to independently add qualitative-temporal constraints with the guarantee that no conflicts arise among the agents. We showed that the plan-decoupling recognition problem in an STTN and the cardinal-minimal plan-decoupling problem are both intractable. However, despite the added expressiveness of the task network, the computational complexity for the plan-decoupling problems do not increase in comparison with their counterparts for qualitative task networks.

Second, we extended the coordination approach used for planning (i.e., to guarantee the existence of a joint dispatching) to multi-agent task-based scheduling. For that purpose, we defined schedule coordination that enables agents to independently construct local schedules (i.e., to add quantitative-temporal constraints) with the guarantee that no conflicts arise among the agents. Additionally, we defined dispatch coordination that enables agents to independently construct local dispatchings with the guarantee that their local dispatchings together is a joint dispatching (i.e., forms a solution to the problem). We found that an STTN is schedule coordinated when it is dispatch coordinated, and vice versa. Therefore, temporal decoupling (Hunsberger, 2002) can be applied (i.e., a coordination mechanism that guarantees that merging the local dispatchings forms a joint dispatching) such that the agents can construct dispatchings for executing their assigned sets of tasks.

Third, we proposed a two-phase approach of autonomous planning followed by autonomous scheduling to enable self-interested agents to separately add planning and scheduling preferences on the execution of the tasks. Here, qualitative-temporal constraints are associated with the planning preferences (or strategic decisions), and the quantitative-temporal constraints are associated with the scheduling preferences (or tactical decisions) of the agents. The a-priori plan and schedule-coordination mechanisms have thus become preference preserving in the sense that (i) the imposed preference constraints do not need to be revised afterwards and (ii) the dispatching satisfies the preference constraints. Because the agents are enabled to make certain non-preferred dispatchings unavailable as a solution for completing the task network, this two-phase approach results in dispatchings that are more highly preferred than a one-shot approach using temporal decoupling.

For future work, it would be interesting to study plan decoupling for certain subclasses of STTNs. First, the factors influencing the computational complexity of plan-decoupling problems in STTNs can be studied (similar to Section 3.2.2). Second, we would like to identify subclasses of STTNs for which the plan-decoupling mechanisms developed for qualitative task networks suffice. This question can be stated as “When is quantitative-temporal really quantitative?” and is related to the work by Cushing et al. (2007) on the question “When is temporal planning really temporal?” Preliminary research (Steenhuisen and Witteveen, 2007a,b) indicates that plan-decoupling mechanisms can be used when only time windows are added to the qualitative task networks, but that this is not possible when task durations are specified as well.
Chapter 5

Empirical Evaluation of using Plan Decoupling

In Chapters 3 and 4, we took a theoretical approach to enabling self-interested agents to independently construct plans and schedules for their own sets of tasks. We assumed that self-interested agents prefer coordination mechanisms that impose a minimum set of additional constraints. The variants of the plan-decoupling problem with minimum coordination sets turned out to be intractable. Therefore, polynomial-time heuristics are needed for constructing—not necessarily minimal—coordination sets, which were shown to exist.

In this chapter, we study the practical consequences for self-interested agents when using decoupling. We do not study the size of the coordination sets themselves nor the construction time required by a decoupling mechanism. For schedule decoupling, we expect self-interested agents to be interested in the effect on the schedule flexibility. This flexibility, however, depends on the (minimal) makespan for completing the task network, which depends on the plan constructed for it. For plan decoupling, self-interested agents will be interested in the effect on the (minimal) makespan itself. Therefore, we will study plan decoupling, and not study schedule decoupling (which has also been studied by Hunsberger (2002)). We focus on two aspects of plan construction by the self-interested planners.

- What is the effect of plan decoupling on the plan-execution cost?
- What is the effect of plan decoupling on the plan-construction time?

We consider two perspectives on these effects of using plan decoupling: the system perspective and the agent perspective. From the system perspective, we are interested in the consequences of using plan decoupling for all agents together. When the system is assumed to have a choice between obedient agents and self-interested agents (that accepted the responsibility for completing their assigned task networks), the following question needs to be answered.

*Given a joint planning task, what are the effects of using plan decoupling for all agents together?*
In particular, we are interested in the price of autonomy (i.e., the ratio of the plan-execution cost with and without plan decoupling) and the speedup (i.e., the ratio of the plan-construction time without and with plan decoupling). In order to quantify these, we compare the performance after applying plan decoupling to the performance achieved by a central planner that constructs a plan for all agents together. For the remainder of this chapter, we will refer to this latter approach as the centralised approach, and to the approach using plan decoupling as the decentralised approach.

From the agent perspective, we are interested in the consequences of using plan decoupling for the agents individually. Here, the agents are assumed to be self interested to start with and, therefore, need to be incentivised to coordinate among each other. These agents need an answer to the following question.

*Given a joint planning task, what are the effects of using plan decoupling for the agents individually?*

For this perspective, we must estimate the price of coordination (i.e., the ratio of the plan-execution cost with and without a coordination mechanism) and the effect of plan decoupling on the plan-construction time. We compare the performance after adding the coordination constraints to the performance achieved with planning without coordination constraints for the uncoordinated subproblem. In fact, these results provide us with upper bounds on the price of coordination, because the joint plans (of all local plans together) constructed without coordination constraints might be inconsistent.

Finally, we consider two types of planning agents: agents that construct sequential plans and agents that construct parallel plans. The plan-execution costs that are minimised by these agents are the total cost (i.e., the sum of all action durations occurring in the plan) and the makespan (i.e., the minimum duration needed for executing the plan), respectively. Therefore, we study the effect of plan decoupling on both metrics.

The outline of this chapter is as follows. In Section 5.1, we consider the system perspective by formulating our hypotheses and giving a detailed description of our experiments, and we present our results in Section 5.2. Thereafter, we shift to the agent perspective for which we give hypotheses and details on the experimental setup in Section 5.3, and we present the results in Section 5.4. In Section 5.5, we conclude by addressing the research questions of this introduction.

## 5.1 Experiments using Plan Decoupling: System Perspective

In this section, we describe our experiments on the costs and benefits of using plan decoupling from a system perspective. In Section 5.1.1, we formulate our hypotheses with regard to the outcome of the experiments. In Section 5.1.2, we present the design of experiments that will be used, and discuss all its dependent
and independent variables. In Section 5.1.3, we present our experimental setup in which we describe the variable instantiations used in the experiments.

5.1.1 Expectations

In this section, we present our hypotheses on the effect of using plan decoupling on the plan-execution cost and on the plan-construction time.

The effect of plan decoupling on plan-execution cost

We consider planning agents that prefer plans with minimal plan-execution cost. In the following hypothesis, we state our expectation on the ratio of the plan-execution cost of the decentralised and centralised approach. For this purpose, we define the price of autonomy as

\[
\text{price of autonomy} = \frac{\text{decentralised plan execution cost}}{\text{centralised plan execution cost}}. \tag{5.1}
\]

In the centralised approach, the planner constructs a joint plan for which it prefers plans with minimal plan-execution cost. A plan-decoupling mechanism does not take the effects on the plan-execution cost into account during the construction of a coordination set. Now, the constructed coordination set is at best compatible with a joint plan of minimal cost. Therefore, in the decentralised approach, the plan-execution cost for an instance is either equal or larger, irrespective of the metric used for plan-execution cost.

**Hypothesis 1.** The price of autonomy is larger than or equal to 1, independent of (i) the size of the problem instance, and (ii) the metric used for plan-execution cost.

The effect of plan decoupling on plan-construction time

Let us start by considering the fact that the problem instances are decomposable into subproblems (i.e., one for each agent). By applying a plan-decoupling mechanism to these decomposed subproblems, these subproblems can be solved independently. Because plan decoupling allows plan construction to be parallelised, we will study the speedup.

\[
\text{speedup} = \frac{\text{centralised plan construction time}}{\text{decentralised plan construction time}}. \tag{5.2}
\]

Suppose that problem instances of size \( n \) have exponential work complexity (i.e., summed plan-construction times of all agents) \( W(n) \approx c^n \) for some \( c > 1 \) in the centralised approach, and that problems are decomposed into \( k > 1 \) equally-sized subproblems. Let us assume that the coordination set has limited size compared to \( n \). Then, the work complexity for the decentralised approach is \( k \cdot W\left(\frac{n}{k}\right) \approx k \cdot c^{\frac{n}{k}} \). Because the exponential term dominates the constant term \( k \) for large values of \( n \), for larger \( n \) we have \( k \cdot c^{\frac{n}{k}} < c^n \).
Hypothesis 2. *Speedup increases with the problem size if the number of subproblems is kept constant.*

We expect speedups larger than 1. However, we might see speedups smaller than 1 for smaller instances, when the reduction plan-construction time does not outweigh the additional cost of the multiple start-up times of all planners together. Differences among the planners can be expected due to implementation details and planning methods, such as start-up times and times spent on pre-processing. In spite of differences among the planners, we are interested in the speedups in plan construction that can be expected.

Continuing the above arithmetic, we can determine the expected speedup by

\[
\frac{W(n)}{W(n/k)} \approx \frac{c^n}{k^{c^2/k}} = \frac{1}{k} c^{(1-\frac{1}{k})n}.
\]

This result shows that exponential speedups can be expected when increasing the instance size, when \(c > 1\) and \(k > 1\) are constant. For smaller values of \(n\) (i.e., smaller problem instances), however, the \(\frac{1}{k}\) term might dominate the exponential term which might result in slowdowns instead.

Hypothesis 3. *When the plan-construction time grows exponentially, the speedup of taking the decentralised approach also grows exponentially.*

Note that both hypotheses above are based on the assumption that the time spent by the planners on the subproblems increases similarly to the work required for constructing a plan for the complete problem. Also note that the problem instance might not always be decomposable into \(k \geq 1\) equally-sized subproblems.

### 5.1.2 Design of Experiments

In multi-agent task-based planning, the problem is to construct a *joint plan* for a given *planning problem instance* as input. This experiment is designed for comparing the *centralised approach* and the *decentralised approach* for solving such planning problems.

For each planning problem instance, we take both the centralised and decentralised approach to constructing a joint plan for it. In the centralised approach, the planning problem instances are solved by letting a single planning agent construct a joint plan from which the plans for each individual agent can be extracted. In the decentralised approach, the planning problem instances are first decomposed into subproblems for each agent, and to which a plan-decoupling mechanism is applied. Thereafter, the plan-decoupled subproblems are given to the individual agents that independently construct plans for these subproblems that can be merged to form the joint plan. In both approaches the same planner is used: by the planning agent in the centralised approach, and by all individual agents in the decentralised approach.

The variables that can be altered in this experiment, called the independent variables, are the following.

**planning problem**

This is the multi-agent task-based planning problem.
number of tasks, number of constraints, and number of agents

For a certain planning problem, these input variables define the task network of the planning problem instance.

plan-decoupling mechanism

This is the mechanism used for plan decoupling the planning problem instance.

planner

This is the software tool used for constructing plans.

The variables that will be measured in this experiment, called the dependent variables, are the following.

plan-construction time

This is the amount of time taken for plan construction by the (planning) agents. In the decentralised, we differentiate work complexity (i.e., summed plan-construction times of all agents) and time complexity (i.e., maximum plan-construction time of all agents).

plan-execution cost

This is the cost required for executing the plan. Here, we differentiate total cost (i.e., the sum of all action durations) and makespan (i.e., the minimum duration needed for executing the plan).

These variables need to be measured. The plan-execution cost can be determined based on the constructed joint plan itself (i.e., offline after the plan has been constructed). The plan-construction time, however, needs to be measured during plan construction (i.e., online during plan construction). Here, special care should be taken that all planners are timed in the same way, and that the same type of time is measured: either the number of CPU cycles used, or the wall clock time.

In Figure 5.1, a sketch is given of the experimental design with its variables. There, the variables are all in italics, with the independent variables having the filled dots and the dependent variables having the open dots.

5.1.3 Experimental Setup

In order to test the hypotheses of Section 5.1.1, we conducted more than 3,000 experiments. We let 11 planners solve 278 planning-problem instances each using the centralised approach, and solving them using the decentralised approach. In each experiment, we measured both the plan-construction time needed by the planner and the plan-execution cost of the constructed joint plan.

We consider two planning problems that have been used in the international planning competition (IPC): Logistics and Depot. The reason for using these problems is that they allow plan decoupling. Both problems are transportation problems, featuring an infrastructure, transportation vehicles, and packages to transport. We use the following sets of planning-problem instances (see Appendix B for a detailed description of the materials and methods).
Logistics-1998
This set has been used in the IPC of 1998 and consists of 30 instances that feature little regularity with respect to the number of tasks, constraints and agents. The number of packages that need to be transported ranges from 4 to 57. Each package requires three tasks $\tau_1, \tau_2, \tau_3$ to be completed in sequence: $\tau_1 \text{ before } \tau_2 \text{ before } \tau_3$. The number of agents ranges from 4 to 48. Here, the number of agents and packages are modified independently, and so do the infrastructure and the number of transportation vehicles used by the agents. For a detailed description of this set, we refer the reader to Table C.1 in Appendix C.

Logistics-2000
This set features 198 instances among which there is much more regularity, based on the number of packages $|O|$ (transportation orders) which ranges from 4 to 100. The number of tasks is $3|O|$ and the number of constraints $2|O|$ having the same task sequences as in Logistics-1998. There are $1 + \lceil \frac{|O|}{3} \rceil$ agents, where the infrastructure and number of transportation vehicles changes only one agent (i.e., increases linearly in $|O|$).

Depot-2010
This set consists of 50 instances with the same infrastructure and number of transportation vehicles, and the number of agents fixed to 23. The number of packages ranges from 1 to 50. The transportation order for each package can be represented by three tasks and two constraints: $\tau_1 \text{ meets } \tau_2 \text{ meets } \tau_3$. 

Figure 5.1: An overview of the experimental design with its variables.
### 5.1. SYSTEM PERSPECTIVE

For each problem, we use a problem-specific plan-decoupling mechanism that is inspired by our Algorithm 1. Our experiments can be classified as either sequential or parallel planning, which differ in the optimisation metric used for the plan-execution costs. While sequential planning minimises the cost of executing the actions, the plans constructed specify a sequence of actions. Because the makespan of a sequential plan is the sum of all action durations, such plans are not used in practise. Therefore, we also use parallel planning in which the planners minimise the makespan.

#### Sequential planning

In these experiments, the plan-execution cost is measured in terms of the *total cost*. For solving the planning problem instances, we use the available planners that competed in the sequential-planning track of the IPC. In Table 5.1, we list the planners used together with the year of participation and rank achieved. Due to the complexity of the problem instances, we do not use optimal planners.

#### Parallel planning

In these experiments, the plan-execution cost is measured in terms of the *makespan*. For solving the planning problem instances, we use the available planners that competed in the so-called temporal-planning track of the IPC. In Table 5.2, we list the planners used together with the year of participation and rank achieved.

<table>
<thead>
<tr>
<th>Name</th>
<th>IPC (rank)</th>
<th>Literature</th>
</tr>
</thead>
<tbody>
<tr>
<td>C3</td>
<td>2008 (4)</td>
<td>Ramírez et al. (2008)</td>
</tr>
<tr>
<td>FF v2.3</td>
<td>2000</td>
<td>Hoffmann (2001)</td>
</tr>
<tr>
<td>FF($h_a$)</td>
<td>2008 (3)</td>
<td>Keyder and Geffner (2008)</td>
</tr>
<tr>
<td>FF($h_{sa}$)</td>
<td>2008 (2)</td>
<td>Keyder and Geffner (2008)</td>
</tr>
<tr>
<td>LPG-td v1.0</td>
<td>2004 (2)</td>
<td></td>
</tr>
<tr>
<td>SGPlan v5</td>
<td>2006 (1)</td>
<td>Chen et al. (2006)</td>
</tr>
<tr>
<td>YAHSP v1.1</td>
<td>2004 (2)</td>
<td>Vidal (2002)</td>
</tr>
</tbody>
</table>

Table 5.1: The planners used for sequential planning.

<table>
<thead>
<tr>
<th>Name</th>
<th>IPC (rank)</th>
<th>Literature</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGPlan v5</td>
<td>2006 (1)</td>
<td>Chen et al. (2006)</td>
</tr>
<tr>
<td>SGPlan v6</td>
<td>2008 (1)</td>
<td>Hsu and Wah (2008)</td>
</tr>
</tbody>
</table>

Table 5.2: The planners used for parallel planning.
5.2 Results using Plan Decoupling: System Perspective

In this section, we present the results for the experiments performed. In Section 5.2.1, we give the experimental results on the effect of using plan decoupling on plan-execution cost. In Section 5.2.2, we show our findings with respect to the effect of using plan decoupling on plan-construction time.

5.2.1 The effect of plan decoupling on plan-execution cost

Regarding the effect of using plan decoupling on the plan-execution cost, we expect that the price of autonomy is larger than or equal to 1, independent of (i) the size of the problem instance, and (ii) the metric used for plan-execution cost (Hypothesis 1). We measured the plan-execution cost for each constructed joint plan and calculated the price of autonomy according to Equation 5.1. To investigate this hypothesis, we present the prices of autonomy for problem instances of increasing size (i.e., sets Logistics-2000 and Depot-2010), for which both sequential and parallel plans were constructed.

Sequential planning

In Figure 5.2, the results are presented for the price of autonomy for sequential planning. Here, the price of autonomy is based on the total cost as the performance metric for plan-execution costs.

For both Logistics-2000 and Depot-2010, the prices of autonomy are bound by a horizontal range from 0.75 to 1.10 for most planners (see Figures 5.2(a) and 5.2(b)).

For Logistics-2000, only the C3 planner (see Figure D.2) has an increasing price of autonomy (i.e., up to 1.60) which is caused by the high-cost plans constructed for the air-traffic subproblems. For Depot-2010, only YAHSP (see Figure D.3) has a decreasing price of autonomy (i.e., down to 0.55), which is explained by the high-cost plans constructed in the centralised approach.

Parallel planning

In Figure 5.3, the changes in plan costs are shown as the price of autonomy for parallel planning. Here, the price of autonomy is based on the makespan as the performance metric for plan-execution costs.

For both sets of problem instances, the prices of autonomy are bound by a horizontal range. This horizontal range seems to be from 0.65 to 1.10 for Logistics-2000 (see Figures 5.3(a)), while being from 0.35 to 1.20 for Depot-2010 (see Figure 5.3(b)).

For Logistics-2000, the results are identical for the two planners used, which can be explained by the fact that these are two subsequent versions of the same planner (i.e., SGPlan). For Depot-2010, SGPlan v5 profits more from using plan decoupling than its successor SGPlan v6. In fact, the plan-execution costs were identical in the centralised approach, but the SGPlan v5 constructed cheaper plans in the decentralised approach.
Figure 5.2: Price of autonomy for sequential planning for increasing number of transportation orders.
Figure 5.3: Price of autonomy for parallel planning for increasing number of transportation orders.
Table 5.3: Price of autonomy (average and standard deviation) for sequential and parallel planning for SGPlan v5 for the sets of problem instances.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>total cost</td>
<td>0.96 (0.04)</td>
<td>0.89 (0.04)</td>
<td>0.94 (0.07)</td>
</tr>
<tr>
<td>makespan</td>
<td>1.07 (0.28)</td>
<td>0.84 (0.14)</td>
<td>0.60 (0.23)</td>
</tr>
</tbody>
</table>

Discussion Concerning Hypothesis 1, the results do not show that the price of autonomy is larger than or equal to 1. In fact, we found that the price of autonomy is planner specific and is often smaller than 1 instead (see Table 5.3). A notable difference between the sequential and parallel planning is the large difference in standard deviation. This difference might be caused by the potentially large effect of the added ordering constraints on the makespan, while this effect is much smaller in terms of the total cost.

One explanation for the results not supporting the hypothesis is based on the type of planners and the plan-decoupling mechanisms used. Assume heuristic planners construct closer-to-optimal plans for the (smaller) subproblem instances than they do for the whole problem instances. Also assume that the plan-decoupling mechanisms used are good heuristics (i.e., for the planning problems considered), that cause the plan-execution cost of the optimal plan not to increase at all (or very little). Under these assumptions, the price of autonomy could actually become smaller than 1 instead. The argument for the hypothesis was based on optimal planners instead of heuristic planners.

5.2.2 The effect of plan decoupling on plan-construction time

In this section, we present our experimental results on the effect of using plan decoupling on the plan-construction time. For this purpose, we measured the plan-construction times required for each joint plan in both centralised and decentralised approach, and calculated the speedup (see Equation 5.2).

We expect that using plan decoupling results in a speedup that increases with the problem size if the number of subproblems is kept constant (Hypothesis 2). Furthermore, we expect that when the plan-construction time grows exponentially, the speedup of taking the decentralised approach also grows exponentially (Hypothesis 3). To investigate these hypotheses, we conducted experiments for problem instances of increasing size (i.e., sets Logistics-2000 and Depot-2010), for which both sequential and parallel plans were constructed.

Sequential planning In Figure 5.4, the results are presented for the speedups for sequential planning on the Logistics-2000 and Depot-2010 problem sets. For Logistics-2000, the speedups seem to converge to planner-specific constants when increasing the number of transportation orders (see Figure 5.4(a)). Although all
planners eventually have speedups larger than 1, some planners (e.g., SGPlan v6) require up to 8 times more time for the smaller instances. These slowdowns are probably caused by the multiple start-up times required in the decentralised approach.

For Depot-2010, the speedups increase rapidly with increasing problem instances, but this increase tapers off (see Figure 5.4(b)). Many planners (e.g., LAMA and SGPlan v6), however, had difficulty solving the problem instances in the decentralised approach. For instance, LAMA already had troubles with small truck subproblems and was not able to solve at least one of the site subproblems for almost all instances, while it outperformed most planners in the centralised approach. The LPG-td planner (see Figure D.1) did not solve any problem larger than 21 transportation orders, and SGPlan v2 (see Figure D.1) already failed at 6 orders.

**Parallel planning** In Figure 5.5, the results are presented for the speedups for parallel planning on the Logistics-2000 and Depot-2010 problem sets.

For Logistics-2000, the plan-construction times are almost identical to their counterparts for sequential planning (see Figure 5.5(a)). Consequently, the speedup graphs are almost identical to those for sequential planning, and seem to converge to a constant.

For Depot-2010, the plan-construction times in the decentralised approach differ, while the results of the centralised do not differ from those for sequential planning. The speedups of both planners seem to converge to constants, which are approximately 10 for SGPlan v5 and 0.01 for SGPlan v6. SGPlan v6 requires much more time in the decentralised approach for problem instances with more than 18 orders, because it kept resetting itself when planning for the truck problems.

**Discussion** Concerning Hypothesis 2, the results do not support that using plan decoupling results in a speedup that increases with the problem size if the number of subproblems is kept constant. Although speedup increases initially, the results for both sequential and parallel planning suggest that the speedups decrease or converge to planner and problem-specific constants. In spite of our efforts, we did not succeed in finding a suitable explanation for these results.

Concerning Hypothesis 3, the results do not support that using plan decoupling in a speedup that grows exponentially, when the plan-construction time grows exponentially. Instead, the results suggest that speedup can converge to planner and problem-specific constants, as in Logistics-2000. Moreover, we saw that keeping the number of subproblems constant might even not result in an increasing speedup at all, as in parallel planning for Depot-2010.

Because both these results are not in line with our expectations, we need to find an explanation. On closer inspection, we discovered that the premise of the hypothesis (that the problem instances are decomposed into $k > 1$ equally-sized subproblems) does not hold for Depot-2010, despite the fact that the number of subproblems was kept constant. We looked at the ratio of time and work complexity (see Fig-
5.2. SYSTEM PERSPECTIVE

Figure 5.4: Speedup for sequential planning for increasing number of transportation orders.
Figure 5.5: Speedup for parallel planning for increasing number of transportation orders.
ures D.4 and D.5) and found out that both Logistics-2000 and Depot-2010 have one subproblem that dominates the other subproblems in terms of plan-construction time. Therefore, we conducted additional experiments in which no such dominating subproblem exists.

In Logistics-1998, no such dominating subproblem exists (see Figure D.6). In Figure 5.6, the speedup results are shown for the sequential planners on the Logistics-1998 instances. Here, the size of the problem instance on the horizontal axis is taken as the logarithm of the required plan-construction time of centralised approach, because the instances of this problem set do not scale in a single dimension. Despite the limited data (i.e., 30 instances), the results suggest a linear relation between the problem size and the logarithm of the speedup. This means that the speedup grows exponentially in the size of the problem.

In conclusion, we found supporting evidence that using plan decoupling results in an exponentially-growing speedup, when the plan-construction times of at least two subproblems grow exponentially (Hypothesis 3).
5.3 Experiments using Plan Decoupling: Agent Perspective

In the previous sections, we looked at the consequences of using plan decoupling from a system perspective. Instead, self-interested agents are more likely to be interested in the consequences of using plan decoupling from their local agent perspective. In this section, we describe how we intend to answer the research questions related to the consequences of using plan decoupling from an agent perspective.

5.3.1 Expectations

We consider self-interested agents that have been assigned a constrained set of tasks. In order to complete their tasks, each agent needs to construct a plan for it. When constructing such local plans independently, the local plans can be in conflict with each other (i.e., the agents are uncoordinated). To guarantee that their independently-constructed plans will result in the completion of their assigned tasks, the agents can coordinate among each other (i.e., using a plan-decoupling mechanism). To study the consequences of using plan decoupling from an agent perspective, we compare uncoordinated and coordinated plan construction performance.

The effect of plan decoupling on plan-execution cost

We consider self-interested planning agents that prefer plans with minimal plan-execution cost. In the following hypotheses, we compare the plan-execution cost of uncoordinated and coordinated planning. For this purpose, we define the price of coordination as

\[
price \text{ of coordination} = \frac{\text{coordinated plan execution cost}}{\text{uncoordinated plan execution cost}}.
\]  

(5.3)

Note that this price of coordination provides us with an upper bound, because the uncoordinated plan-execution cost might be based on local plans of which the joint plan is inconsistent. Because the effect on plan costs is not taken into account when constructing the coordination sets, the coordination constraints might conflict with previous optimal plans (i.e., the optimal plans might not be plans for the coordinated task networks anymore).

Hypothesis 4. The price of coordination is larger than or equal to 1.

However, a difference here can be expected between total cost and makespan, based on the relation between the ordering constraints and the plan-execution cost. Recall that the total cost is defined as the (weighted) sum of all actions, and the makespan is defined as the (minimum) time required for executing the partially-ordered actions. While the ordering constraints of the coordination set have a direct—potentially negative—impact on the makespan, the ordering of the actions
is not relevant when the total cost is considered. The total cost only increases when the ordering constraints require additional actions. Hence, we get the following hypothesis.

**Hypothesis 5.** *The addition of coordination constraints results in a larger increase in costs in terms of makespan than in terms of total cost.*

The effect of plan decoupling on plan-construction time

We consider consistent planning-problem instances for which there exists a plan that satisfies all constraints. After the addition of the set of coordination constraints, the coordinated instances remain consistent. Therefore, plans can be constructed for both the uncoordinated and coordinated subproblems, such that the speedup can be calculated.

\[
\text{speedup} = \frac{\text{uncoordinated plan construction time}}{\text{coordinated plan construction time}}.
\] (5.4)

From the literature (Mitchell et al., 1992), we know that adding constraints can lead to both speedups and slowdowns for 3SAT. There, speedups were reported for adding constraints (clauses) to an instance that already was unsatisfiable. However, adding constraints to a satisfiable instance, such that the instance remains satisfiable, was shown never to result in a speedup and often in a slowdown.

Planning-problem instances can be translated into 3SAT-problem instances. Because we consider consistent planning-problem instances (i.e., there exists a plan for it), they can be translated into a satisfiable 3SAT-problem instance. Plan decoupling the planning-problem instance results in a set of coordination constraints being added to the consistent task network (associated with the planning-problem instance), such that the task network remains consistent. In fact, these coordination constraints are added to the agents’ sub-task networks. Therefore, we expect similar results for plan decoupling as those reported for adding constraints to satisfiable 3SAT instances: That planning after plan decoupling requires more plan-construction time compared to uncoordinated planning.

**Hypothesis 6.** *From an agent perspective, plan decoupling results in speedup values smaller than or equal to 1, independent of (i) the size of the problem instance, and (ii) the metric used for plan-execution cost.*

### 5.3.2 Design of Experiments

In multi-agent task-based planning, a *planning problem instance* is decomposed such that each *agent* is assigned its own *subproblem*. Now, each agent is responsible for constructing a *local plan* for its subproblem.

We decompose each planning problem instance into subproblems for the agents. For each subproblem, we consider both the *uncoordinated* (i.e., without coordination constraints) and the *coordinated* (i.e., after plan decoupling) task networks. Then,
we let each agent independently construct a local plan for their subproblem both with and without coordination constraints (i.e., compare performance on the coordinated versus their uncoordinated counterparts).

In Figure 5.7, a visual overview is given of the variables and methods used of this experiment. The independent and dependent variables are equal to those described in Section 5.1.2 for the experiments on using plan decoupling from a system perspective.

![Figure 5.7: An overview of the variables and methods.](image)

### 5.3.3 Experimental Setup

In order to test the hypotheses of Section 5.3.1, we conducted more than 4,000 experiments using the design of experiments of Section 5.3.2. We let 14 planners solve 308 subproblems, each using the uncoordinated and the coordinated variants. In each experiment, we measured both the plan-construction time needed by the planner and the plan-execution cost of the constructed joint plan.

The subproblems are based on 30 instances from the Logistics planning problem (see Section 5.1.3), which is a transportation problem. Specifically, we use the following set of planning problem instances.

**Logistics-1998**

This set has been used in the IPC of 1998 and consists of 30 instances, involve 4 to 48 agents, that need to construct plans for transporting 4 to 57 packages. Decomposing these 30 instances results in 308 subproblems. Because all ordering constraints among the tasks are inter-agent constraints, there are no ordering constraints in the uncoordinated subproblems. In each subproblem,
the infrastructure interconnects the locations in a city by means of an all-to-all network. For a detailed description of this set, we refer the reader to Table C.1 in Appendix C.

For plan decoupling, we used the same problem-specific plan-decoupling mechanism as used in Section 5.1.3 for Logistics. We conduct the following two experiments.

**Sequential planning** In this experiment, the plan-execution cost is measured in terms of the total cost. In addition to the planners listed in Table 5.1, we used the planners listed in Table 5.4.

**Parallel planning** In this experiment, the plan-execution cost is measured in terms of the makespan. We used the same set of planners as used in the experiments on the consequences on using plan decoupling from a system perspective (see Table 5.2).

### 5.4 Results using Plan Decoupling: Agent Perspective

In this section, we present the results for the experiments performed. In Section 5.4.1, we give the experimental results on the effect of using plan decoupling on plan-execution cost. In Section 5.4.2, we show our findings with respect to the effect of using plan decoupling on plan-construction time.

#### 5.4.1 The effect of plan decoupling on plan-execution cost

Regarding the effect of using plan decoupling on the plan-execution cost, we expect the price of coordination to be larger than or equal to 1 (Hypothesis 4). We conducted experiments with both sequential and parallel planning for which we found the following results. Comparing these types of planning, we expect the price of coordination to be higher in terms of makespan than in terms of total cost (Hypothesis 5).

In order to investigate these hypotheses, we measured the plan-execution cost for each constructed joint plan and calculated the price of coordination according to Equation 5.3. We present the prices of coordination relative to the plan-execution cost of the plan constructed with uncoordinated planning.

<table>
<thead>
<tr>
<th>Name</th>
<th>IPC (rank)</th>
<th>Literature</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTG-Plan</td>
<td>2008 (6)</td>
<td>Huang et al. (2008)</td>
</tr>
<tr>
<td>HSPF</td>
<td>2008 (2)</td>
<td>Haslum (2008)</td>
</tr>
<tr>
<td>HSP0</td>
<td>2008 (3)</td>
<td>Haslum (2008)</td>
</tr>
</tbody>
</table>

Table 5.4: Overview of additional planners used.
Sequential planning  In Figure 5.8, the results are shown of the price of coordination for sequential planning on the subproblems for the city agents of the Logistics-1998 set. Here, the price of coordination is based on the total cost as the performance metric for plan-execution costs.

From the results for the optimal planner HSPF, it can be observed that the plan-execution cost of the optimal plan increases for a few instances. Moreover, the largest increase in plan-execution cost involved only one additional action, which occurred only once. In fact, the optimal plan of only 5 (out of 308) subproblem instances increased.

Concerning the heuristic planners, the price of coordination differs among the planners with prices that can be smaller and larger than 1. On the one hand, SGPlan v6 and LPG-td speed (see Figure D.7) constructed plans with much lower plan-execution costs when using plan decoupling. These results are explained by the fact that these planners construct plans with relatively high plan-execution costs for the uncoordinated instances. On the other hand, SGPlan v5 and YAHSP v1.1 already constructed plans with low plan-execution costs for the uncoordinated instances. For these planners, using plan decoupling results in an increase in the plan-execution costs.
Parallel planning In Figure 5.9, the results are given for the price of coordination for parallel planning on the subproblems for the city agents of the Logistics-1998 set. Here, the price of coordination is based on the makespan as the performance metric for plan-execution costs.

![Figure 5.9: Price of coordination for parallel planning, relative to the plan-execution cost resulting from uncoordinated planning.](image)

The results are identical for the two planners used, which can be explained by the fact that these are two subsequent versions of the same planner (i.e., SGPlan). The price of coordination is in the range 0.50 to 2.50. The plan-execution cost decreased for only 8 out of 308 problem instances, when using plan decoupling.

Discussion Concerning Hypothesis 4, the results suggest that the price of coordination is larger than or equal to 1 for parallel planning and optimal sequential planning, as expected. However, both decreasing and increasing plan-execution costs were found for the heuristic planners used for sequential planning. The planners with a price of coordination that is smaller than 1 construct plans with relatively high plan-execution costs for the uncoordinated instances, while constructing plans with costs that are similar to the other planners when using plan decoupling. Therefore, it seems that the plan-decoupling mechanism is a good heuristic for Logistics.

Concerning Hypothesis 5, the results (i.e., comparing the price of coordination for sequential and parallel planning) seem to suggest that the plan-execution cost in terms of makespan is larger than that in terms of total cost, as expected. However, only two planners were used for parallel planning. Therefore, additional experiments are needed (i.e., for parallel planning) for investigating this hypothesis.
5.4.2 The effect of plan decoupling on plan-construction time

In this section, we present our experimental results on the effect of using plan decoupling on the plan-construction time. We expect that coordinated planning (using plan decoupling) requires more plan-construction time compared to uncoordinated planning (Hypothesis 6). To investigate this hypothesis, we measured the plan-construction times required for each plan for both uncoordinated and coordinated planning, and calculated the speedup (see Equation 5.4).

**Sequential planning** In Figure 5.10, the speedup results are presented for sequential planning on the subproblems for the city agents of the Logistics-1998 set. These results show a neutral effect (i.e., a speedup close to 1) for the optimal planner HSPF, while another optimal planner HSP0 (see Figure D.8) showed some speedup. The results show speedups for heuristic planners such as FF v2.3 and SGPlan v5.

**Parallel planning** For parallel planning, the speedup results (see Figure D.9) are neutral (i.e., a speedup close to 1) for the tested problem instances and planners. But, it should again be noted that data from only two planners was available (i.e., SGPlan v5 and v6) for parallel planning.
Discussion The results do not support Hypothesis 6. In fact, the results even seem to suggest the opposite, which is that coordinated planning requires less plan-construction time compared to uncoordinated planning. Moreover, the results for neither optimal nor for heuristic planners support Hypothesis 6.

The reasoning behind Hypothesis 6 was based on the possibility of translating a planning-problem instance into a 3SAT instance. We assumed that adding constraints to the planning-problem instance would have the same effect as adding constraints to the related 3SAT instance. This assumption does not seem to hold. A possible explanation to this is that the coordinated planning-problem instance is not translated into the same 3SAT instance with some additional clauses, but into a different 3SAT instance instead.

5.5 Summary and Conclusion

In this chapter, we wanted to quantify the costs and benefits for the agents of using plan decoupling. Instead of looking at (the construction of) the coordination sets, we focused on the effect of plan decoupling on the plan-execution cost and the plan-construction time. More specifically, we wanted to study these effects from both a system perspective and an agent perspective in order to estimate the potential costs and benefits when providing a set of agents with autonomy and coordinating with other agents.

From a system perspective, we found the price of autonomy to be close and often even smaller than 1 on average (i.e., the decentralised approach resulted in cheaper plans) for both sequential (i.e., total cost) and parallel planning (i.e., makespan), independent of the size of the problem instance. For the speedup, we found values larger than 1 that can converge to planner-specific constants or even increase exponentially, when increasing the size of the problem instance.

From an agent perspective, we found the price of coordination to be around 1 (i.e., both larger and smaller values) for (heuristic) sequential planning, and to be larger than or equal to 1 for (heuristic) parallel planning. These results seem to indicate that—from an agent perspective—the price of coordination for parallel planning is higher than for sequential planning. For the speedup, we found values larger than 1 for heuristic sequential planners, and values around 1 (i.e., both larger and smaller values) for optimal planners.

From our experiments, we learned that some of the results were better than expected. This difference is explained by the fact that our expectations were based on optimal planners, while we used heuristic planners in the experiments. When using an optimal planner, the joint plan of all constructed plans for all subproblems together cannot have a lower plan-execution cost than the plan constructed for the problem instance. However, it turns out that heuristic planners profit from decomposing a problem instance and letting them solve the set of subproblems, because they are able to construct closer-to-optimal plans when the problem instances are smaller.
In short, our experiments with (heuristic) sequential planners provide the following results. We found that using plan decoupling results in similar or smaller plan-execution cost from both system and agent perspective. Concerning the plan-construction time, we mainly found speedups larger than 1 from both system and agent perspective. Because we used only two planners in our experiments in (heuristic) parallel planning, our results are preliminary only.
Chapter 6

Conclusions and Research Directions

The goal of this thesis was to enable a set of autonomous agents to construct a global solution for a planning or scheduling problem by coordinating among the agents such that the independently constructed locally-consistent plans or schedules together form such a global solution. We focused on self-interested agents which are—from an autonomy point of view—the most challenging type of agents.

We considered the construction of plans and schedules separately in two consecutive phases. Then, we allowed the agents to add qualitative-temporal and quantitative-temporal constraints (i.e., reflecting their planning and scheduling preferences, respectively) independently from the other agents. Because self-interested agents do not want to revise their independently-constructed plans or schedules, we took an a-priori coordination-by-design approach, called decoupling. Self-interested agents prefer coordination at the lowest possible cost, which we assumed to be reflected by the size of the coordination set (i.e., the number of coordination constraints).

Our approach generalises the available literature on plan and schedule decoupling. First, it extends the applicability of plan decoupling to frameworks with richer types of qualitative-temporal constraints. Second, it generalises schedule decoupling such that the agents are allowed to add a less restrictive type of scheduling constraints. Finally, it provides an integrated approach to coordinated planning and scheduling for autonomous agents, because it combines both decoupling problems in a unifying framework (i.e., with qualitative and quantitative-temporal constraints).

6.1 Answer to Research Questions

In this section, we will evaluate the extent to which the contributions of this thesis provide answers to our research questions as posed in Section 2.5. Besides answering the question by briefly summarising our main results, we will provide directions for future work.
Research question 1
Is it possible to extend the applicability of plan decoupling to more expressive frameworks with other temporal constraints, besides simple precedence constraints?

We wanted to study plan-coordination problems in frameworks with rich types of planning constraints among a set of tasks: qualitative- and quantitative-temporal constraints. Here, we wanted to consider coordinated planning for sets of tasks with ordering and synchronisation constraints among them as the qualitative-temporal constraints. Concerning the quantitative-temporal constraints, we wanted to allow the duration of a task, the time window in which a task should start and/or finish, and to allow minimum and maximum separation times between the tasks to be specified. In Chapter 1, we already provided examples of the plan-coordination problems that arise when allowing such constraints.

To allow such rich types of planning constraints, we extended the existing plan-coordination framework (i.e., with precedence constraints only) in two ways. First, we defined a qualitative task network as an assigned set of tasks which end points are constrained by precedence and synchronisation relations. In these qualitative task networks, we identified the plan-coordination problem that arises (i.e., when allowing all agents to construct locally-consistent plans) and proved that this plan-coordination problem can be solved by plan decoupling. Second, we defined a quantitative task network that extends the qualitative-task network to allow time windows, task durations, and task separation times. Again, we identified the plan-coordination problem that arises and proved that this plan-coordination problem can be solved by plan decoupling. Hence, we increased the applicability of plan decoupling to task networks with richer qualitative-temporal constraints (e.g., synchronisation constraints), and quantitative-temporal constraints (e.g., time windows and task durations). We, thereby, have made plan decoupling available for problems that require these richer types of constraints.

This result demonstrates the flexibility of the framework for adding new types of constraints. For future work, we suggest to study decoupling in even more expressive frameworks with additional temporal and non-temporal constraints. For instance, the decoupling approach could be applied to distributed constraint systems in general (van der Hoek et al., 2011).

Research question 2
For plan decoupling, what are the consequences in terms of the computational complexity of adding different types of temporal constraints to the framework?

In answering this question, we studied the computational complexity of plan decoupling a framework with qualitative-temporal or quantitative-temporal constraints (i.e., the qualitative and quantitative task networks, respectively).

For qualitative task networks, we defined two variants of the plan-decoupling problem that either construct a subset-minimal or a cardinal-minimal coordination
set. We proved that both these variants are intractable in general. We showed that
the complexity of plan decoupling in qualitative task networks (i.e., with precedence
and synchronisation constraints) does not increase in comparison with plan decou-
pling for task networks with precedence constraints only. Moreover, we studied the
computational complexity of subclasses of the plan-decoupling problem where either
the number of agents or the number of tasks per agent is limited. We proved that
these subclasses remain intractable, even when the problem instances are limited to
consist of few agents, or tasks per agent.

For quantitative task networks, we defined and studied the cardinal-minimal
plan-decoupling problem. We proved this plan-decoupling problem to be intractable
in general. We showed that—from a complexity point of view—plan decoupling in
quantitative task networks is not harder than for qualitative task networks in spite
of the increased expressivity of the framework.

Hence, we proved that the computational complexity of the plan-decoupling
problem does not increase when adding synchronisation constraints, time windows,
task durations, and task separation times to the framework. As a consequence, we
can enable self-interested agents to construct their local plans independently from
the other agents in these richer frameworks without the plan-decoupling problem
becoming (computationally) harder.

The reason that the complexity of plan decoupling does not increase is that
neither the framework itself nor the interaction with the framework require more
complex problems to be solved. Both consistency checking and constraint propa-
gation can be done in polynomial time for both qualitative and quantitative task
networks. Therefore, the complexity of plan decoupling does not differ between
qualitative and quantitative task networks.

For future work, the influence of the underlying problems on the complexity of
plan decoupling itself should be studied. For instance, determining the effect of
the complexity of consistency checking on the complexity of plan decoupling would
enable us to determine which constraints can be added without increasing the com-
plexity of plan decoupling. A start could be to determine the effect of allowing
multiple time-window constraints (as used in temporal constraint satisfaction prob-
lems (Dechter et al., 1991)) which cause consistency checking to become intractable.
Furthermore, the complexity of other subclasses could be determined based on the
type of agent-dependency graph (e.g., line, star, or tree shaped). These studies gain
insight in the influence of the structure of the problem instances on the complexity
of plan-decoupling problems. We did some preliminary work on line and star-shaped
agent-dependency graphs (Steenhuisen et al., 2007, 2008).

Research question 3

What is the difference between plan and schedule decoupling (in a quantitative
framework)?

Both plan and schedule decoupling guarantee the existence of a joint dispatching
that satisfies all imposed preference constraints. However, these types of decoupling
differ in at least two ways.
First, plan and schedule decoupling differ in the types of constraints the agents are allowed to specify. In the planning phase, qualitative-temporal constraints are associated with the agents’ planning preferences (or strategic decisions). In the scheduling phase, quantitative-temporal constraints are associated with the scheduling preferences (or tactical decisions) of the agents. As such, plan and schedule decoupling coordinate among the agents at different levels of decision making (i.e., the strategic and tactical level, respectively).

Second, schedule decoupling is more restricting than plan decoupling. Schedule decoupling enables agents to add quantitative-temporal constraints, which constraints subsume the qualitative-temporal constraints. Therefore, the agents are also enabled to add qualitative-temporal constraints, which makes them plan coordinated. Hence, schedule coordination implies plan coordination, while the reverse does not hold.

This result demonstrates the possibility and advantage of having a separate planning and a scheduling phase. Our work generalises both plan coordination (Valk, 2005) and temporal decoupling (Hunsberger, 2002). On the one hand, we extended the applicability of the preference-preserving plan coordination to richer frameworks. On the other hand, we generalised dispatching-preserving temporal decoupling to allow the agents to add scheduling constraints (cf. constructing a dispatching).

For future work, the robustness of plan and schedule-decoupling solution should be studied in dynamic environments. In such environments, coordination is not guaranteed when changes occur (e.g., when new tasks are allocated or agents can violate constraints). For instance, when an agent fails to satisfy a scheduling constraint (i.e., possibly a schedule-coordination constraint), this does not necessarily violate a planning constraint as well. It should be studied when coordination sets can be repaired and when they must be newly constructed for maintaining plan and schedule coordination.

**Research question 4**

How does the computational complexity of plan decoupling compare to that of schedule decoupling (in a quantitative framework)?

We already proved plan decoupling to be intractable, and saw that plan and schedule decoupling are strongly related. In fact, we saw that plan coordination can be achieved by solving the schedule-decoupling problem. Therefore, we wanted to compare the complexity of plan decoupling with that of (preference-preserving) schedule decoupling.

First, we defined schedule and dispatch coordination for, respectively, the preference- and dispatching-preserving variants of schedule decoupling. Before determining the complexity, we studied the relation between these variants of schedule decoupling. We found that the dispatching-preserving schedule-decoupling problem coincides with dispatching-preserving schedule-decoupling problem, also called temporal decoupling (Hunsberger, 2002). Therefore, schedule coordination can be achieved by temporal decoupling (Hunsberger, 2002), of which the cardinal-minimal
variant is polynomially solvable (Planken et al., 2010). Hence, we found that the complexity of plan and schedule decoupling differs (i.e., intractable versus tractable).

As a consequence, we can derive heuristics for constructing a plan-coordination set from the algorithms for solving the schedule-decoupling problem. Such heuristics are needed due to the intractability of the plan-decoupling problem. The plan-coordination set can easily be constructed by determining the qualitative-temporal constraints implied by the schedule-coordination set.

The difference in complexity between plan and schedule decoupling can be attributed to the constraints that the agents might add. Note that these types of constraints determine the potential conflicts that can arise among the agents. Because the preference-preserving decoupling problems are to prevent these potential conflicts, their complexity depends on the constraints the agents are allowed to add.

For future work, the applicability of plan and schedule decoupling should be extended to a richer quantitative framework (e.g., to allow disjunctions of time-window constraints). In such richer frameworks, the complexity of basic problems (e.g., consistency checking) increases. We expect the differences in complexity between plan and schedule decoupling to decrease.

**Research question 5**

What is the effect of using plan decoupling on the plan-execution costs and plan-construction times for all agents together?

Because self-interested agents need an incentive for using a plan-decoupling mechanism, we wanted to estimate the practical consequences of using plan decoupling. When the system is assumed to have a choice between obedient agents and self-interested agents (that accepted the responsibility for completing their assigned task networks), we are interested in the consequences of using decoupling for all agents together. We did not want to focus on the effect of schedule decoupling, because the effect on schedule flexibility (which we expect the agents to be interested in) has already been studied by Hunsberger (2002) and depends on the makespan (i.e., the shortest time from start to finish). Instead, we expect self-interested agents to be interested in the effect on the (minimal) makespan itself, which depends on the constructed plans. Therefore, we focused on the consequences of plan decoupling on the plan-execution costs and plan-construction times.

Instead of studying the size of the coordination sets or the construction time required by the plan-decoupling mechanism, we studied the effect of plan decoupling on the plan-execution costs and plan-construction times for the agents. We considered two logistics planning problems, and used planners that either construct sequential or parallel plans. We determined the price of autonomy (i.e., the ratio of the plan-execution cost with and without plan decoupling) and the speedup (i.e., the ratio of the plan-construction time without and with plan decoupling).

We found the price of autonomy to be close to and often even smaller than 1 on average (i.e., lower plan-execution costs when using plan decoupling) for both sequential planning (i.e., total cost) and parallel planning (i.e., makespan), independent of the size of the problem instance. For the speedup, we found values larger
than 1 that can either converge to planner-specific constants or increase exponentially, when increasing the size of the problem instance.

From our experiments, we learned that the results for sequential planners were better than expected. This difference is explained by the fact that our expectations were based on optimal planners, while we used heuristic planners in the experiments. When using an optimal planner, the joint plan after plan decoupling (of all constructed plans for all subproblems together) cannot have a lower plan-execution cost than the plan constructed for the complete problem instance. However, it turns out that heuristic planners profit from plan decoupling a problem instance and letting them solve the set of subproblems, because they are able to construct closer-to-optimal plans when the problem instances are smaller.

**Research question 6**

What is the effect of using plan decoupling on the plan-execution costs and plan-construction times for the agents individually?

When all agents together are incentivised to use a plan-decoupling mechanism, there is still no guarantee that all agents individually are willing to participate. When dealing with self-interested agents, each agent individually needs an incentive. Therefore, we are interested in the consequences of using plan decoupling for the agents individually.

We studied the effects of the additional coordination constraints on the plan-execution costs and plan-construction times for the agents individually. We considered a logistics planning problem, and used planners that either construct sequential or parallel plans. We compared planning by a single agent after adding the coordination constraints to the performance achieved with planning without coordination constraints for the uncoordinated instances. For each run of the experiment, we determined the price of coordination (i.e., the ratio of the plan-execution cost with and without a coordination mechanism) and the speedup (i.e., the ratio of the plan-construction time without and with plan decoupling). In fact, these results provide us with upper bounds on the price of coordination, because the joint plans (of all local plans together) constructed without coordination constraints might be inconsistent.

We found the price of coordination mostly to be larger than or equal to 1 for both parallel planning and optimal sequential planning. For heuristic sequential planning, however, we also found the price of coordination to be smaller than 1 for some planners. These planners (with a price of coordination that is smaller than 1) construct plans with relatively high plan-execution costs for the uncoordinated instances, while constructing plans with costs that are similar to the other planners when using plan decoupling. Furthermore, the results seem to suggest that the plan-execution cost in terms of makespan is larger than that in terms of total cost. However, only two planners were used for parallel planning. Therefore, additional experiments are needed (i.e., for parallel planning) for investigating this hypothesis.

In the considered planning problems, sequential planning agents are incentivised to using plan decoupling both from a system and agent perspective. Although this
result is promising, it does not mean that this result can be translated to any other planning problem. It should, however, be noted that the heuristics used in these experiments were not optimised for any performance metric (i.e., nor for the number of actions, nor for the minimum makespan). Therefore, results might even improve by optimising the heuristic.

For parallel planners, the results gave no incentive for using plan decoupling. In relation to scheduling, however, the effect of plan decoupling on the allowed minimum-makespan schedules is important. Due to the limited number of parallel planners used, the results on the effect of plan decoupling on makespan are preliminary.

In future work, the effects of plan decoupling on parallel planning should be studied thoroughly. Here, the heuristic should be adapted to take efficiency of the allowed plans into account as well (i.e., instead of consistency only). In addition to the problem instances used for our experiments, problem instances should be used that constrain tasks with time windows. These issues are specifically important in ground handling and container transshipment (see Chapter 1).
Appendix A
Planning Preliminaries

Before moving on to the experiments themselves, we will provide some preliminaries on planning because it is used as the experimental environment. First, we will explain how planning problems and instances are specified. Thereafter, we will describe the different types of planners for solving these instances and two cost metrics for their constructed plans.

A.1 Planning Problems

In 1998, the first International Planning Competition (IPC) was organised, hosted by the International Conference on Artificial Intelligence Planning Systems (AIPS-1998), to provide a platform for empirical evaluation and comparison of automated problem-independent planners (Long et al., 2000). Since then, it has been organised every two years and has grown out to consist of three parts: (i) the deterministic part that considers deterministic and fully-observable environments (i.e., classical planning), (ii) the uncertainty part, which considers nondeterministic and probabilistic actions, and (iii) the learning part, where the planners are first allowed to construct problem-dependent knowledge in an offline training period. In our experiments, we look at problems from the deterministic part, and therefore continue with the description of that part only.

In the deterministic part, the participating planners need to solve a collection of instances in various problems which are specified in a formal language: the Planning Domain Definition Language (PDDL) (Ghallab et al., 1998). PDDL defines a planning problems in terms of predicates that define the states the world can be in, and the actions that define the possible state changes by means of value changes of a subset of predicates. Over time, the language evolved with a new version of PDDL with almost each IPC (Fox and Long, 2001, 2003, 2006; Gerivini and Long, 2006; Hoffmann and Edelkamp, 2005). These versions added expressiveness to the basic language, including durative actions and temporal constraints.

In Figure A.1, the PDDL specification is given of an example problem. Such a PDDL problem-specification consists of 5 parts:
(define (domain Depot)
  (:requirements :strips :typing)
  (:types place locatable - object
    depot distributor - place
    truck hoist surface - locatable
    pallet crate - surface)
  (:predicates (at ?x - locatable ?y - place)
    (on ?x - crate ?y - surface)
    (in ?x - crate ?y - truck)
    (lifting ?x - hoist ?y - crate)
    (available ?x - hoist)
    (clear ?x - surface))

  (:action LIFT
                   (on ?y ?z) (clear ?y))
    :effect (and (not (at ?y ?p)) (lifting ?x ?y) (not (clear ?y))
              (not (available ?x)) (clear ?z) (not (on ?y ?z)))
  )

  (:action DROP
    :effect (and (available ?x) (not (lifting ?x ?y)) (at ?y ?p)
              (not (clear ?z)) (clear ?y) (on ?y ?z))
  )

  (:action LOAD
    :effect (and (not (lifting ?x ?y)) (in ?y ?z) (available ?x))
  )

  (:action UNLOAD
    :effect (and (not (in ?y ?z)) (not (available ?x))
              (lifting ?x ?y))
  )

  (:action DRIVE
    :parameters (?x - truck ?y - place ?z - place)
    :precondition (and (at ?x ?y))
    :effect (and (not (at ?x ?y)) (at ?x ?z))
  )
)

Figure A.1: Specification of an example problem in PDDL.
domain
In this part, the problem name is specified for referencing purposes.

:requirements
Here, the features (e.g., :typing and :durative-actions) are listed that are used in this problem.

:types
This part gives a (hierarchical) set of types of which objects can be instantiated and used in the problem.

:predicates
This lists all predicates that can be specified on the different types of objects in the problem instances.

:action
In this part, actions are defined in terms of preconditions and effects (i.e., in terms of valuations of a set of predicates immediately before and after the execution of this action).

Example A.1. In Figure A.1, a description is given in PDDL of an example transportation problem called Depot. Consider the action DRIVE. Because the action involves driving a truck from one place to another (i.e., either a depot or a distributor), this action takes three parameters. The precondition states that the predicate (at ?x ?y) must be true, which represents that the truck ?x is at the starting place ?y. The effect of applying the action is that (not (at ?x ?y)) and (at ?x ?z) have become true, which means that truck ?x is not at place ?y but at place ?z.

A problem description specifies a planning problem, but not the instances themselves. In a problem instance, the terminology (i.e., :types and :predicates) is used from the problem description for specifying the initial state and the set of goals. Such a problem instance consists of 5 parts (e.g., see Figure A.2):

problem
Here, the name of this problem instance is given for referencing.

:domain
This specifies the problem of which this is an instance.

:objects
In this part, all objects are listed that are used, and are the only objects on which predicates can be defined in this instance.

:init
In this part, a list of all positive ground predicates is given. All other ground predicates are assumed to be false, resulting in a complete description of the initial state.

:goal
Here, a conjunction of predicates is given (both positive and negative ground predicates) which define the set of goal states.
A.2 Planners and Plan Costs

The competitors in the deterministic part of the IPC are so-called problem-independent planners, which are programmed for the purpose of competing in one (or more) of the three track categories: (i) sequential, (ii) temporal, and (iii) net benefit. For our experiments, we are interested in the former two categories. These categories differ in the requirements for the planners and in the performance metric (i.e., objective) by which the constructed plans are judged. Each track category has a satisficing and an optimisation track in which the planners can compete.

- **Sequential satisficing/optimisation**
  INSTANCE: A PDDL with action costs
  OBJECTIVE: minimise total cost

- **Temporal satisficing/optimisation**
  INSTANCE: A PDDL with durative actions
  OBJECTIVE: minimise makespan

The plan cost can be defined as the total cost, which is the sum of all action durations, or as the makespan, which is the total time needed for executing the plan. Note that, in the sequential tracks, it suffices to construct a plan that can be executed sequentially, because improving the plan’s makespan doesn’t change the total cost. In the temporal tracks, on the other hand, the (partially-)overlapping execution of multiple actions needs to be taken into account for constructing the
A.2. PLANNERS AND PLAN COSTS

plan with the shortest time from begin to end. Note that a shorter makespan might involve more actions (i.e., a more expensive plan in terms of the total cost).

We conclude by explaining the scoring system used, which is relevant because the planners are programmed for this competition. Each planner is given a limited amount of time (e.g., 30 minutes in IPC-2008) for each problem instance. The scoring is based on the plan costs (i.e., exact plan-construction time does not count towards scoring). For each solved problem instance, 1 point is awarded to the planner(s) with the best plan while the others get the ratio of the best plan costs to its plan cost. The difference between the satisficing and optimisation track is that planners in the optimisation track get no points for the complete problem (i.e., not just that problem instance) when reporting at least one sub-optimal plan.
Appendix B

Materials and Methods

In this section, we describe the material used for conducting the experiments. First, we describe the planning problems, the translation of a problem instance to its task-network representation, and illustrate the result of applying our plan-decoupling heuristic. Thereafter, the planners will briefly be described together with the specification of the computer systems they were run on.

B.1 Planning Problems

In the experiments, two planning problems are used as our experimental environments, called Logistics and Depot. These problems are chosen due to the applicability of our plan-decoupling approach. In fact, the Logistics problem was already used by Valk (2005) in his experiments on plan coordination. The choice for the Depot problem is due to its similarity to Logistics, and the fact that it requires synchronised actions by multiple agents. In this section, we describe both these planning problems.

B.1.1 Logistics Problem

The Logistics problem is a transportation problem in which packages must be transported from their initial locations to their respective destinations. For this transportation task, there are two modalities that can be used: trucks and aeroplanes. Both trucks and planes can carry any amount of packages.

The infrastructure consists of multiple cities, that are interconnected by air bridges. Each city $c_i$ consists of multiple locations $L_i = \{l_{i,j} \mid j = 1, 2, \ldots, m_i\}$ that are connected by a fully-connected graph of roads. These roads can only be traversed by trucks, which makes them responsible for all intra-city transport. In each city $c_i$, one location $l_{i,1}$ is an airport which is connected to the other airports in other cities by a fully-connected graph of air bridges. These air bridges can only be used by the planes, which makes them responsible for all inter-city transport.

The set of transportation orders is given by $O \subseteq L \times L$ with $L = \bigcup_{i=1}^{n} L_i$ being the set of all locations of all cities. A transportation order $o = (l, l')$ specifies that
a package is initially located at some location $l$ and needs to be transported to its destination location $l'$. In the given infrastructure, this results in two types of transportation orders: An intra-city order is an order when both locations are in the same city, and an inter-city order when the locations are in different cities.

**Example B.1.** In Figure B.1, an instance is shown of the Logistics problem. There are 3 cities $cty1$, $cty2$, and $cty3$, consisting of respectively 2, 3, and 4 locations with one truck in $cty2$ and $cty3$, both located at the respective airports, and 2 trucks in $cty1$ that are both not located at the airport. There are 2 aeroplanes: one is located in $cty1$ and another in $cty3$. In this infrastructure, 4 transportation orders must be completed: (i) pkg1 from 112 to 134, (ii) pkg2 from 134 to 132, (iii) pkg3 from 121 to 112, and (iv) pkg4 from 112 to 123.

The Logistics problem has been used in the international planning competition in both 1998 and 2000. In Appendix C, the PDDL specification of this problem is given. From that description, we see that the Logistics problem instances need to be solved by selecting and ordering the actions LOAD, UNLOAD, and MOVE for both trucks and planes.

**B.1.2 Depot Problem**

The Depot problem is a transportation problem in which crates must be moved between sites (i.e., depots and distributors) while these crates are stacked at those sites. Besides transporting crates between the sites, stacking problems need to be solved at these sites. In fact, this Depot problem is a combination of the Logistics problem with the well-known BlocksWorld problem.
Remark B.2. The BlocksWorld problem is a classical planning problem, where the problem is to achieve a specified stacking of the blocks from an initial situation. This can be achieved by unstacking the blocks to the table or stacking them on top of each other, using a given number of arms (i.e., with a hand that can at most hold a single block at a time).

The transportation infrastructure consists of an interconnected set of sites \( S = \{s_1, s_2, \ldots, s_n\} \), where each site \( s_i \) has a non-empty set \( P_i = \{p_{i,j} \mid j = 1, 2, \ldots, m_i\} \) of pallets. At each site \( s_i \), there is a non-zero number of hoists \( h_i \) and a possibly-empty set of trucks. Each hoist can (un)stack crates on one of the \( m_i \) stacks available (i.e., a crate can only be stacked on a pallet or on top of another crate), and load/unload crates onto/from a truck when present. There is no stacking limit (i.e., stacks of crates can be of arbitrary height), and all stacks and trucks are directly reachable for all hoists. The sites are connected by a fully-connected graph of roads over which the trucks can move, which makes them responsible for inter-site transportation. While the trucks can carry any amount of crates, the hoists can only lift one crate at a time.

A transportation order specifies a crate to be moved from its initial place in a stack at a site, to a position in a stack at some site. Similar to the orders in the Logistics problem, we differentiate intra and inter-site orders depending on whether the initial and destination site differ.

Example B.3. In Figure B.2, an instance is shown of the Depot problem. There are 3 sites \( \text{dpt0} \), \( \text{dst0} \), and \( \text{dst1} \), each having one pallet and one hoist available. There are 2 trucks which are both located at site \( \text{dpt0} \). Furthermore, there are 4 crates: each pallet has one crate stacked onto it, except for the pallet at \( \text{dst1} \) which has a stack of 2 crates. In fact, the depicted situation is the initial state of the example instance of Figure A.2.

This Depot problem was used in the international planning competition of 2002. Therefore, a PDDL specification is available for this problem, which in fact was already given in Figure A.1. From that problem definition, we read, besides the...
self-explanatory types and predicates, that there are 5 actions \textsc{Lift}, \textsc{Drop}, \textsc{Load}, \textsc{Unload}, and \textsc{Drive} for changing the initial state into a goal state. It is up to the planners to select and order these actions.

\section*{B.2 Task Network Representation}

The problem instances are described in PDDL, which is in terms of states (i.e., as sets of literals) and operators (i.e., actions) for transforming one state into another. This representation is not a task-network formulation (as used in this thesis) which is required for applying plan decoupling. Therefore, we need to translate these instances from their PDDL description into their task-network representation such that we can apply plan decoupling. This requires us to address the following issues: (i) What are the tasks? (ii) What are the constraints? (iii) What are the agents? and (iv) What is the task assignment?

Here, we will describe how these problem instances can be represented as task networks by using the problem descriptions given in the previous section. The basic idea is to decompose each transportation order into a task chain (i.e., a sequence of tasks) and to assign each task to an agent that is capable of completing it with its vehicles. In Logistics, for instance, these orders can be decomposed into tasks for transportation by either truck or plane. Then, by choosing the agents in such a way that they plan for the sets of vehicles that are given tasks to complete, the task assignment is trivially solved (e.g., one agent per city).

\subsection*{B.2.1 Logistics Problem}

A Logistics problem instance \((L, O)\) can be translated into its task-network representation as follows. An intra-city order can be translated into a single elementary task that can be completed by a single agent (i.e., an agent responsible for transportation within a single city). Because an inter-city order \(o = (l_{i,p}, l_{j,d}) \in O\) requires transportation by multiple agents, we need to consider how such an inter-city order can be decomposed. Due to the infrastructure, each inter-city transportation order \(o\) consists of a sequence \(o^{\text{pre}} \text{ before } o^{\text{in}} \text{ before } o^{\text{post}}\) of intra and inter-city sub orders.

Recall from the problem description that both cities \(c_i, c_j\) have one airport (i.e., locations \(l_{i,1}\) and \(l_{j,1}\), respectively). Therefore, these sub orders can all be specified as the transportation orders \(o^{\text{pre}} = (l_{i,p}, l_{i,1}), o^{\text{in}} = (l_{i,1}, l_{j,1}), \) and \(o^{\text{post}} = (l_{j,1}, l_{j,d})\). By associating tasks with these sub orders, we construct a task sequence \(\tau^{\text{pre}} \text{ before } \tau^{\text{in}} \text{ before } \tau^{\text{post}}\) for completing each inter-city transportation order.\footnote{Note that the intra-city parts of an inter-city transportation order can be trivial (e.g., when a package is initially located at the airport). For every such trivial intra-city order, we also define a task \(\tau\) for it.}

Now, we need to define what the agents are in Logistics. We define an agent for each region in which it has a set of vehicles (i.e., trucks or planes). We do not...
B.2. Task Network Representation

Figure B.3: Task-network representation of a Logistics instance.

have a separate agent for each vehicle, because then their individual responsibilities would be unclear due to the remaining task-assignment problem of tasks over these vehicle agents. Hence, we define an agent for each region (i.e., for each city and one for air traffic) such that each elementary task (i.e., the intra or inter-city part of a task) can only be assigned to one agent. Now, all tasks $\tau_{\text{pre}}$, $\tau_{\text{post}}$ are assigned to city agents $A_i$ together with the intra-city tasks $\tau$, and all tasks $\tau_{\text{in}}$ are assigned to the air-traffic agent. In Figure B.3, the task-network representation is given for the problem instance of Example B.1.

B.2.2 Depot Problem

The Depot instances can be translated to their task-network representations in a similar way as the Logistics instances. Similarly, the transportation orders can be decomposed into a sequence of intra and inter-site orders.

For completing those orders, we define three types of elementary tasks: a task $\tau_{c_{\text{pre}}}$ for the loading of a crate $c$ onto a truck, a task $\tau_{c_{\text{in}}}$ for transporting this crate from its pickup site to its destination site, and a task $\tau_{c_{\text{post}}}$ for stacking this crate as specified in the goal. Note here that this Depot problem requires synchronisation due to the (un)loading of the trucks by the hoists being a synchronised activity. Therefore, we get the following task chains $\tau_{c_{\text{pre}}} \text{ meets } \tau_{c_{\text{in}}} \text{ meets } \tau_{c_{\text{post}}}$. Further note that additional constraints might hold among the task chains due to the crates being stacked. For instance, when two crates are stacked on top of each other, the crate at the bottom of the stack can only be lifted when the top crate has been lifted previously.

Finally, we define agents for each site (i.e., depot or distributor) and one that plans for the inter-site transportation tasks. In this way, the hoists and trucks have become the resources that are available to the agent to complete its tasks. Due to this choice of the agents, the task-assignment problem has become trivial, because there is only one possible assignment of tasks to agents. Now, we can represent the problem instance of Figure A.2 as the task network shown in Figure B.4.

Note the similarity of the task network in Figure B.4 with the task network of
the Logistics instance in Figure B.3. The main difference is that the inter-agent constraints are precedence constraints in the Logistics problem, while the Depot problem requires the use of synchronisation constraints.

B.3 Plan decoupling

Given the example task-network representation of the problem instances, it is easy to see that the joint plans are not guaranteed to be feasible when local plans are constructed by the agents independently. Therefore, we will use the depth-partitioning algorithm (see Algorithm 1) for plan decoupling these instances. In this section, we describe the set of additional precedence constraints due to plan decoupling, and how we specify them in PDDL. The problem with PDDL is that it does not allow precedence constraints to be defined directly.

B.3.1 Logistics Problem

In Logistics, the depth-partitioning algorithm assigns a depth \( d = 1 \) to all pre-flight transportation tasks, and sets all post-flight transportation tasks to depth \( d = 3 \). As a consequence, an ordering constraint \( \tau_i^{\text{pre}} \text{ before } \tau_j^{\text{post}} \) is added for every pair of tasks that are assigned to the same agent and belong to different orders \( o_i, o_j \). For the air-traffic agent, all tasks get depth \( d = 2 \) which results in no constraints being added. The depth-partitioning algorithm now guarantees that all plans constructed by the city agents and air-traffic agent can be joined together to constitute a feasible joint plan for the task network and, thus, the original planning problem.

Example B.4. Consider the situation at city cty1 in Figure B.3. Here, the depth-partitioning algorithm has added the following constraints: \( t_{\text{pkg1}}^{\text{pre}} \prec t_{\text{pkg3}}^{\text{post}} \) and \( t_{\text{pkg4}}^{\text{pre}} \prec t_{\text{pkg3}}^{\text{post}} \).

Let us now turn to representing these precedence constraints in PDDL. In Logistics, the coordination constraints enforce that each city agent completes its set
B.3. PLAN DECOUPLING

Example B.5. Consider the precedence constraints added to the agent of city \textit{cty}1 in Figure B.3, as described in Example B.4. The constraints $t_{\text{pre},crt2}^{\text{pre}} < t_{\text{post},crt0}^{\text{post}}$ and $t_{\text{pre},crt1}^{\text{pre}} < t_{\text{post},crt0}^{\text{post}}$ are encoded in the action \textsc{LOAD-UNLOAD} by means of its preconditions and effects, as shown in Figure B.5.

B.3.2 Depot Problem

In Depot, applying the depth-partitioning algorithm results in a plan decoupling that closely resembles the situation in Logistics. All tasks of loading outgoing crates into trucks must precede all the tasks of unloading incoming crates, which corresponds to the ordering of pre and post-flight tasks. Because the (un)loading tasks require synchronisation between the site and truck agent, every set of synchronised activities between any pair of agents needs to be totally ordered. As a heuristic, we construct the coordination sets based on the order in which the crates are stacked (i.e., for loading tasks) and need to be stacked (i.e., for unloading tasks). Note that this heuristic is specific for this problem.

Example B.6. Consider the situation at distributor \textit{dst}1 in Figure B.4, where there are two loading tasks for \textit{crt}1 and \textit{crt}2. Due to \textit{crt}2 being stacked on \textit{crt}1 (see Figure A.2), the constraint that forces \textit{crt}2 to be loaded before \textit{crt}1 seems to be much more intuitive than the other one. The described heuristic results in the following constraints: $t_{\text{pre},crt2}^{\text{pre}} < t_{\text{pre},crt1}^{\text{pre}} < t_{\text{post},crt0}^{\text{post}}$. 

\begin{verbatim}
{:action LOAD-UNLOAD
 :parameters (?y - plane ?z - airport)
 :precondition (and (not (used DoneLoadUnload)) (at ?y ?z)
 (at pkg1 ?z) (at pkg4 ?z) (in pkg3 ?y))
 :effect (and (used DoneLoadUnload) (not (at pkg1 ?z)) (in pkg1 ?y)
 (not (at pkg4 ?z)) (in pkg4 ?y)
 (not (in pkg3 ?y)) (at pkg3 ?z))
}
\end{verbatim}

Figure B.5: Action for guaranteeing the coordination constraints on the pre and post-flight tasks at city1.

of pre-flight tasks before starting any of its post-flight tasks. Instead of translating each precedence constraint separately, we will encode the ordering of these two sets of tasks. We add an action \textsc{LOAD-UNLOAD} to the problem description which has the precondition that these pre-flight tasks are completed (i.e., the involved packages are at the airport), and has the effect that the packages involved in the post-flight tasks are at their initial location (i.e., the airport). To guarantee that this action has its desired effect we add a literal \textit{DoneLoadUnload} at each city that needs to be made true. This literal can only be made true by means of this action, and can only be executed once because it also requires \textit{DoneLoadUnload} to be false. The only way to get the packages of the post-flight tasks at their initial locations is through the use of this \textsc{Load-Unload} action (i.e., we removed the actions related to planes).
APPENDIX B. MATERIALS AND METHODS

(:action LOAD-UNLOAD
 :parameters (?z - truck ?p - place)
 (on crt2 IOPallet) (on crt1 crt2) (clear crt1) (in crt0 ?z)))
 :effect (and (used DoneLoadUnload) (not (at crt2 ?p)) (in crt2 ?z)
 (not (at crt1 ?p)) (in crt1 ?z) (not (on crt2 IOPallet))
 (not (on crt1 crt2)) (not (clear crt1)) (not (in crt0 ?z))
 (at crt0 ?p) (on crt0 IOPallet) (clear crt0))
)

Figure B.6: Action for guaranteeing the coordination constraints on the load and unload tasks at distributor1.

Let us now turn to representing these precedence constraints in PDDL. For Depot, the approach taken for the Logistics problem (i.e., constraining one set of tasks to precede another set) is insufficient, because a total ordering needs to be specified on the set of synchronised tasks per pair of agents. To encode this total ordering, we add an action LOAD-UNLOAD with the preconditions and effects that encode the coordination constraints. Here, we make use of the fact that a specific stack of crates can only be achieved by stacking them in that specific order (i.e., a precedence order). To this end, we introduce an IOPallet at each site which will be used for encoding the total ordering of load and unload tasks. This pallet is initially empty and is used to mimic the loading and unloading of crates, the order of which can be defined by means of the stacking of the crates on this pallet.

Example B.7. Consider the precedence constraints added to the agent of distributor dst1 in Figure B.4, as described in Example B.6. The coordination constraints \( t^\text{pre.e}_{crt2} < t^\text{pre.e}_{crt1} < t^\text{post.s}_{crt0} \) are encoded by mimicking the (un)loading of crates through the (un)stacking of these crates on an IOPallet. To force that crt2 is loaded before crt1, we require that crt2 is stacked onto IOPallet before crt1 (i.e., stack crt1 on top of crt2). This prerequisite is encoded in the LOAD-UNLOAD by means of its preconditions, as shown in Figure B.6. In the effect of this action, the unload task becomes available by replacing all loaded crates by crate crt0 which needs to be unloaded.

Parallel-planning instances

In the planning problem instances considered so far no task durations were specified. While this specification suffices for sequential planning, it does not for parallel planning. However, these problem instances can be used for parallel planning by specifying durations for the actions the problem file (e.g., :duration (= ?duration 10)).

For Logistics, we let (un)loading for both trucks and planes take 1, driving takes 2 time units, and flying has a duration of 3. For Depot, we defined lift and drop to take 1, loading to take 3, unloading to take 4 time units, and driving to have a duration of 10 units.
B.4 Measuring Methods

Finally, we need to describe our measuring methods and the runtime environment. As described in Section 5.1, we are interested in the plan costs and the (total) plan-construction time.

Concerning the plan costs, we use the number of actions and the makespan for the sequential and parallel planning-problem instances, respectively. These performance metrics are well defined, such that the plan costs of each constructed plan can be determined unambiguously afterwards by analysing the output files.

Although the planners also report a plan-construction time, it is unclear what exactly is timed and whether these values are unbiased. Therefore, we use the `time` command-line tool for measuring the time used by the invoked planner, while all experiments ran on identical machines and under the same conditions (e.g., same hardware and operating system).

Unfortunately, this approach could not be applied to the anytime planners (i.e., DAE, DTG, LAMA, and TFD), because they do not stop after they have constructed their plan. Anytime planners keep constructing plans of decreasing plan costs, until they are stopped or cannot improve on the best constructed plan. Therefore, we use their first constructed plan.

We set the time limit to 20 hours, which is the maximum plan-construction time to be used by the planners for any problem instance. However, we did stop some planners earlier when they took much longer than what was reasonably to be expected. This was when they took orders of magnitude longer than expected based on the results of other planners (e.g., after 10 hours while all other planners took at most 10 seconds), at which point they were assumed not to output any plan.

B.4.1 Runtime Environment

All experiments ran on otherwise-idle Fedora Linux workstations in our labrooms, which are all identical and contain two quad-core Intel® Xeon® CPUs running at 2.33 GHz, having 4 MByte of cache, 16 GByte of RAM, and 50 GByte available disk space. We did not restrict the planners in making use of the available resources of the machine.

\(^2\)This planner is partially-based on evolutionary algorithms, with their own stopping criterion. However, it does construct multiple plans and overwrites the previous result when it finds an improved plan. As such, we qualify it as an anytime planner.
Appendix C

Benchmark Set Details

(define (problem example) (:domain Logistics)
    (:objects cty1 cty2 cty3 - city
     l12 l22 l23 l32 l33 l34 - location
     l11 l21 l31 - airport
     trk1 trk2 trk3 trk4 - truck
     pln1 pln2 - airplane
     pkg1 pkg2 pkg3 pkg4 - package)

    (:init
     (in-city l11 cty1) (in-city l12 cty1)
     (in-city l12 cty2) (in-city l22 cty2) (in-city l23 cty2)
     (in-city l31 cty3) (in-city l32 cty3) (in-city l33 cty3) (in-city l34 cty3)
     (at trk1 l12) (at trk2 l12) (at trk3 l21) (at trk4 l31)
     (at pln1 l11) (at pln2 l31)
     (at pkg1 l12) (at pkg2 l34) (at pkg3 l21) (at pkg l12)
    )

    (:goal (and (at pkg1 l34)
                 (at pkg2 l32)
                 (at pkg3 l12))
           (at pkg4 l23))
)

Figure C.1: PDDL specification of the Logistics example instance.
Figure C.2: PDDL specification of the Logistics problem.
<table>
<thead>
<tr>
<th>instance</th>
<th>cities</th>
<th>locations per city</th>
<th>trucks</th>
<th>planes</th>
<th>packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>prob1</td>
<td>6</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>prob2</td>
<td>10</td>
<td>2</td>
<td>10</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>prob3</td>
<td>14</td>
<td>3</td>
<td>14</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>prob4</td>
<td>13</td>
<td>4</td>
<td>23</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>prob5</td>
<td>9</td>
<td>2</td>
<td>11</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>prob6</td>
<td>18</td>
<td>4</td>
<td>26</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>prob7</td>
<td>11</td>
<td>2</td>
<td>11</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>prob8</td>
<td>20</td>
<td>3</td>
<td>33</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>prob9</td>
<td>11</td>
<td>6</td>
<td>28</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>prob10</td>
<td>23</td>
<td>2</td>
<td>23</td>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>prob11</td>
<td>3</td>
<td>3</td>
<td>22</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>prob12</td>
<td>27</td>
<td>4</td>
<td>55</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>prob13</td>
<td>29</td>
<td>4</td>
<td>29</td>
<td>9</td>
<td>23</td>
</tr>
<tr>
<td>prob14</td>
<td>11</td>
<td>2</td>
<td>47</td>
<td>8</td>
<td>29</td>
</tr>
<tr>
<td>prob15</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>23</td>
</tr>
<tr>
<td>prob16</td>
<td>11</td>
<td>3</td>
<td>52</td>
<td>13</td>
<td>7</td>
</tr>
<tr>
<td>prob17</td>
<td>4</td>
<td>3</td>
<td>45</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>prob18</td>
<td>30</td>
<td>3</td>
<td>30</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>prob19</td>
<td>8</td>
<td>8</td>
<td>30</td>
<td>11</td>
<td>19</td>
</tr>
<tr>
<td>prob20</td>
<td>15</td>
<td>12</td>
<td>19</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>prob21</td>
<td>14</td>
<td>9</td>
<td>40</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>prob22</td>
<td>30</td>
<td>4</td>
<td>74</td>
<td>6</td>
<td>44</td>
</tr>
<tr>
<td>prob23</td>
<td>8</td>
<td>2</td>
<td>28</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>prob24</td>
<td>9</td>
<td>12</td>
<td>54</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>prob25</td>
<td>27</td>
<td>13</td>
<td>29</td>
<td>13</td>
<td>18</td>
</tr>
<tr>
<td>prob26</td>
<td>47</td>
<td>2</td>
<td>106</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td>prob27</td>
<td>15</td>
<td>14</td>
<td>28</td>
<td>9</td>
<td>57</td>
</tr>
<tr>
<td>prob28</td>
<td>20</td>
<td>17</td>
<td>83</td>
<td>5</td>
<td>42</td>
</tr>
<tr>
<td>prob29</td>
<td>27</td>
<td>3</td>
<td>46</td>
<td>10</td>
<td>53</td>
</tr>
<tr>
<td>prob30</td>
<td>22</td>
<td>12</td>
<td>80</td>
<td>3</td>
<td>15</td>
</tr>
</tbody>
</table>

Table C.1: Detailed description of the Logistics-1998 instances.
Appendix D

Additional Figures

Figure D.1: Speedup for sequential planning for increasing number of transportation orders (Depot-2010).
Figure D.2: Price of autonomy for sequential planning for increasing number of transportation orders (Logistics-2000).

Figure D.3: Price of autonomy for sequential planning for increasing number of transportation orders (Depot-2010).
Figure D.4: Time complexity as a ratio of the work complexity for decentralised sequential planning, relative to the work complexity.
Figure D.5: Time complexity as a ratio of the work complexity for decentralised parallel planning, relative to the work complexity.
Figure D.6: Time complexity as a ratio of the work complexity for decentralised sequential planning on the Logistics-1998 instances, relative to the work complexity.

Figure D.7: Price of coordination for sequential planning, relative to the plan-execution cost resulting from uncoordinated planning (Logistics-1998).
Figure D.8: Speedup results for sequential planning, relative to the plan-construction time required for uncoordinated planning (Logistics-1998).

Figure D.9: Speedup results for parallel planning, relative to the plan-construction time required for uncoordinated planning (Logistics-1998).
Bibliography


Acknowledgements

Getting my PhD has taken some years of my life, years I will never forget. My thanks go out to all people that supported me during my PhD research. I thank Cees, my daily supervisor and promotor, for the many things he taught me and the numerous discussions we had. To Cees and Thomas, I say thanks for your patience, for all reviews of my writing, and for all the constructive comments on structuring the thesis. I thank all my colleagues and former colleagues at the Algorithmics group for many different reasons. Mathijs had told me that I would have the experience of a life time. It has been: thanks!

I also wish to thank all my colleagues at SIKS for all enjoyable courses, and all colleagues around the globe that I met and talked with on all these conferences and workshops.

After almost three years of research, I became a father with the birth of Mart. Besides being congratulated, people told me I would have a hard time completing my thesis. In due time, we also welcomed Carien into our family, and people thought it was over. Although the schedule was changed (i.e., completing the thesis mid 2009 has somewhat been delayed), the plan of obtaining my PhD remained. In fact, these last words are written with Jorne (our youngest) sitting on my lap. These tasks could not have been completed with the support and gentle kick-in-the-butt of Anke for which I am very grateful. Finally, I also wish to thank my family and friends for their support and patience throughout.
Acknowledgements
Summary

Coordinated Multi-Agent Planning and Scheduling

In this thesis, we study coordination problems that arise when autonomous agents need to construct a joint plan or a joint schedule for achieving a joint goal. These problems arise when a set of agents each with its own capabilities have a (common) goal, which goal cannot be achieved by any single agent alone. Here, plans and schedules need to be constructed for all agents that are not in conflict with each other, and together achieve the goal.

Application areas include—but are not limited to—disaster rescue, aircraft ground handling at airports and container transshipment at seaports.

Because the agents pursue a common goal, they dependent on each other due to constraints between the tasks assigned to different agents. These inter-agent dependencies must be taken into account when constructing plans and schedules, called coordinated planning and scheduling.

The central problem of this thesis is how to reconcile the autonomy of the agents with coordinated planning and scheduling.

In Chapter 1, we look at two scenarios where self-interested agents need to construct plans and schedules for their assigned sets of tasks. We illustrate the inter-agent conflicts that can arise when the agents construct their plans or schedules independently from each other. We also show two ways of coordinating the agents: By tightening the time windows within which tasks must be executed, and by adding ordering constraints among the tasks.

In Chapter 2, we describe how a joint goal can be decomposed into a constrained set of tasks, called a task network, such that completing all tasks—satisfying all constraints—achieves the joint goal. We assume such a decomposition to be given together with a task assignment such that each agent is capable of and responsible for completing its set of tasks. We describe the available coordination mechanisms that solve plan or schedule-coordination problems either a-priori, a-interiori, or a-posteriori with respect to local plan or schedule construction. Whether coordination at that moment is acceptable depends on the agent’s level of autonomy.

We focus on self-interested planning and scheduling agents, because these agents maintain the highest level of autonomy. When a coordination mechanism is ac-
ceptable for self-interested agents, then it is—from an autonomy point of view—acceptable for agents at any level of autonomy. Self-interested agents pose the hardest constraints on solving the coordination problems.

Self-interested agents require that they can specify their planning and scheduling preferences independently from the other agents, and that they do not need to alter these preferences. Hence, preference-preserving plan and schedule-coordination mechanisms are needed that allow agents to specify their preference constraints and guarantee the existence of a dispatching that satisfies all preference constraints. Therefore, only a-priori coordination-by-design approaches are acceptable.

By casting a set of additional constraints, called a coordination set, onto the task network, agents can be coordinated for independent planning and scheduling. We use plan decoupling and schedule decoupling to denote the preference-preserving coordination mechanisms for the planning and scheduling phase, respectively.

We found a gap in the existing literature concerning coordination mechanisms for self-interested planning and scheduling agents. On the one hand, plan decoupling cannot yet be applied when tasks are constrained by constraints other than precedence constraints (e.g., synchronisation constraints and time windows). Schedule decoupling, on the other hand, cannot yet be used for a less restrictive type of scheduling than dispatching (i.e., it is dispatching preserving instead of preference preserving). For bridging this gap, we formulate the following research questions.

**Research question 1:** Is it possible to extend the applicability of plan decoupling to more expressive frameworks with other temporal constraints, besides simple precedence constraints?

Expanding the possibilities of applying plan decoupling to richer temporal frameworks requires us to revisit the computational complexity of its associated problems.

**Research question 2:** For plan decoupling, what are the consequences in terms of the computational complexity of adding different types of temporal constraints to the framework?

A task network with qualitative-temporal constraints (e.g., precedence constraints) can be embedded in a task network with quantitative-temporal constraints (e.g., a simple temporal network). By using this relation, we can compare plan and schedule decoupling and address the following questions.

**Research question 3:** What is the difference between plan and schedule decoupling (in a quantitative framework)?

If there is a difference between these two preference-preserving approaches, we are interested in the difference in computational complexity.

**Research question 4:** How does the computational complexity of plan decoupling compare to that of schedule decoupling (in a quantitative framework)?
Besides the theoretically-motivated questions above, we are interested in the practical consequences of using decoupling. After all, it must be in the agent’s interest to apply a decoupling approach (i.e., the costs should not outweigh the benefits), when the agents are self interested. Here, we make a distinction between the effects of using plan decoupling for all agents together and the effect for an agent individually.

**Research question 5:** What is the effect of using plan decoupling on the plan-execution costs and plan-construction times for all agents together?

**Research question 6:** What is the effect of using plan decoupling on the plan-execution costs and plan-construction times for the agents individually?

In Chapter 3, we study plan decoupling of a set of tasks constrained by qualitative-temporal constraints, called a qualitative task network. We define these qualitative task networks for representing an assigned set of primitive tasks, being constrained by precedence and synchronisation constraints.

In these qualitative task networks, we show the plan-coordination problem that arises when allowing all agents to construct locally-consistent plans. Two variants of the plan-decoupling problem are defined for constructing either a subset-minimal or a cardinal-minimal coordination set. We prove that the decision problems for both variants are intractable in general. We show that the complexity of the problems does not increase when allowing synchronisation and precedence constraints in comparison with allowing only precedence constraints. We also study the complexity of subclasses where the number of agents or the number of tasks per agent is limited, and prove that these subclasses remain intractable. Finally, we were able to formulate a polynomial-time heuristic for constructing a plan decoupling, for which we provide a proof of correctness.

In Chapter 4, we extend the qualitative task networks of Chapter 3 such that quantitative-temporal constraints can be specified (e.g., time windows and task durations). In these simple temporal task networks (STTNs), two coordination problems arise, which we call the plan-coordination problem and the schedule-coordination problem. The plan-coordination problem arises when locally-consistent plans together might not allow a dispatching (e.g., when the precedence relation contains a cycle). The schedule-coordination problem arises when locally-consistent schedules might not be compatible with each other (i.e., there exists no dispatching that satisfies both). We study plan- and schedule decoupling for solving the coordination problems that arise.

First, we extend the definitions of plan-decoupling problems and study their computational complexity. We show that the plan-decoupling recognition problem in an STTN and the cardinal-minimal plan-decoupling problem are both intractable. Despite the added expressiveness of the task network, the computational complexity for the plan-decoupling problems do not increase in comparison with their counterparts for qualitative task networks.
Second, we extend the coordination approach used for planning (i.e., to guarantee the existence of a joint dispatching) to multi-agent task-based scheduling. For that purpose, we defined *schedule coordination* that enables agents to independently construct local schedules (i.e., to add quantitative-temporal constraints) with the guarantee that no conflicts arise among the agents. Additionally, we defined *dispatch coordination* that enables agents to independently construct local dispatchings with the guarantee that their local dispatchings together is a joint dispatching (i.e., forms a solution to the problem). We found that an STTN is schedule coordinated when it is dispatch coordinated, and vice versa. Therefore, temporal decoupling (Hunsberger, 2002) can be applied (i.e., a coordination mechanism that guarantees that merging the local dispatchings forms a joint dispatching) such that the agents can construct dispatchings for executing their assigned sets of tasks.

In Chapter 5, we study the practical consequences for self-interested agents when using decoupling. Instead of looking at (the construction of) the coordination sets, we focus on the effect of plan decoupling on the plan-execution cost and the plan-construction time. More specifically, we study these effects from both a *system perspective* and an *agent perspective* in order to estimate the potential costs and benefits when providing a set of agents with autonomy and coordinating with other agents.

From a system perspective, we found the *price of autonomy* (i.e., the ratio of the plan-execution cost with and without plan decoupling) to be close and often even smaller than 1 on average (i.e., the decentralised approach resulted in cheaper plans) for both sequential (i.e., total cost) and parallel planning (i.e., makespan), independent of the size of the problem instance. For the *speedup* (i.e., the ratio of the plan-construction time without and with plan decoupling), we found values larger than 1 that can converge to planner-specific constants or even increase exponentially, when increasing the size of the problem instance.

From an agent perspective, we found the *price of coordination* (i.e., the ratio of the plan-execution cost with and without a coordination mechanism) to be around 1 (i.e., both larger and smaller values) for (heuristic) sequential planning, and to be larger than or equal to 1 for (heuristic) parallel planning. These results seem to indicate that—from an agent perspective—the price of coordination for parallel planning is higher than for sequential planning. For the speedup, we found values larger than 1 for heuristic sequential planners, and values around 1 (i.e., both larger and smaller values) for optimal planners.

In Chapter 6, we present the main conclusions of this thesis. Research question 1 can be answered positively, since we extended the applicability of plan decoupling to task networks with richer qualitative-temporal constraints (e.g., synchronisation constraints), and quantitative-temporal constraints (e.g., time windows and task durations). This result demonstrates the flexibility of the framework for adding new types of constraints. The answer to research question 2 is that the computational complexity of plan decoupling does not change for the temporal constraints considered. Concerning research question 3, plan and schedule decoupling differ in the types of coordination problems they solve, because the potential conflicts differ when
adding planning or scheduling preferences (i.e., qualitative-temporal or quantitative-temporal constraints). Because schedule decoupling is more restricting than plan decoupling, schedule coordination implies plan coordination, while the reverse does not hold. For research question 4, we found that the complexity of plan and schedule decoupling differs (i.e., intractable versus tractable). Concerning the effect of using plan decoupling from a system perspective (research question 5), we found the price of autonomy to be close and often even smaller than 1 on average (i.e., the decentralised approach resulted in cheaper plans) for both sequential (i.e., total cost) and parallel planning (i.e., makespan), independent of the size of the problem instance. For the speedup, we found values larger than 1 that can converge to planner-specific constants or even increase exponentially, when increasing the size of the problem instance. Concerning the effect of using plan decoupling from an agent perspective (research question 6), we found the price of coordination mostly to be larger than or equal to 1 for (heuristic) sequential planning, and to be larger than or equal to 1 for (heuristic) parallel planning. For the speedup, we found values larger than 1 for heuristic sequential planners, and values around 1 (i.e., both larger and smaller values) for optimal planners.

In this thesis, we successfully used a two-phase approach of autonomous planning followed by autonomous scheduling to enable self-interested agents to separately add planning and scheduling preferences on the execution of the tasks. We associate qualitative-temporal constraints with the planning preferences (or strategic decisions), and quantitative-temporal constraints are associated with the scheduling preferences (or tactical decisions) of the agents. The a-priori plan and schedule-coordination mechanisms have thus become preference preserving in the sense that (i) the imposed preference constraints do not need to be revised afterwards and (ii) the dispatching satisfies the preference constraints.
Samenvatting

Gecoördineerd plannen en roosteren voor en door meerdere agenten

In dit proefschrift bestuderen we de coördinatie problemen die zich voordoen als 
autonome agenten een gezamenlijk plan of een gezamenlijk rooster moeten bouwen
voor het bereiken van een gezamenlijke doel. Deze problemen ontstaan wanneer
een aantal agenten, elk met zijn eigen mogelijkheden, een (gemeenschappelijke) doel
hebben, welk doel niet kan worden bereikt door één agent alleen. Dan moeten
plannen en roosters worden gebouwd voor alle agenten, die niet in strijd zijn met
elkaar en samen het doel bereiken.

De toepassingsgebieden omvatten—maar zijn niet beperkt tot—noodhulp,
grondafhandeling op luchthavens en containeroverslag in zeehavens.

Omdat de agenten een gemeenschappelijk doel nastreven zijn ze afhankelijk van
elkaar als gevolg van restricties tussen taken, die toegewezen zijn aan verschillende
agenten. Bij de bouw van plannen en roosters moet rekening gehouden worden met
deerse interagent afhankelijkheden, het zogeheten gecoördineerde plannen en roosteren.

Het centrale probleem van dit proefschrift is hoe de autonomie van de de agenten
met gecoördineerd plannen en roosteren te verzoenen.

In hoofdstuk 1 gaan we in op twee scenario’s waar zelfzuchtige agenten plannen
en roosters moeten bouwen voor de hun toegewezen sets van taken. We illustreren
de interagent conflicten die kunnen ontstaan als de agenten hun plannen of roosters
onafhankelijk van elkaar bouwen. We tonen ook twee manieren om de agenten te
coördineren: Door het verkleinen van de tijdsvensters waarbinnen taken uitgevoerd
moeten worden en door het toevoegen van volgorde restricties tussen de taken.

In hoofdstuk 2 beschrijven we hoe een gezamenlijk doel kan worden opgesplitst
in een aantal taken met restricties, een taaknetwerk, zodat met het voltooien van
al deze taken—binnen de gestelde restricties—het gezamenlijke doel bereikt wordt.
We beschouwen een dergelijke opsplitsing als gegeven samen met een toekenning van
deze taken aan agenten die elk in staat zijn tot en verantwoordelijk voor het voltooien
van deze taken. We beschrijven de beschikbare coördinatiemechanismen die plan-
of roostercoördinatieproblemen oplossen hetzij a-priori, a-interiori of a-posteriori
met betrekking tot het bouwen van de lokale plannen of roosters. Welke vorm van
coördinatie acceptabel is hangt af van het niveau van de agent van autonomie.
Wij richten ons op zelfzuchtig plannende en roosterende agenten, omdat deze agenten de hoogste mate van autonomie willen behouden. Wanneer een coördinatiemechanisme aanvaardbaar is voor zelfzuchtige agenten, dan is het—vanuit een autonomie oogpunt—aanvaardbaar voor agenten op elk niveau van autonomie. Zelfzuchtige agenten leggen de hoogste eisen op voor het oplossen van coördinatieproblemen.

Zelfzuchtige agenten vereisen dat ze hun voorkeuren voor plannen en roosters onafhankelijk van de andere agenten kunnen specificeren, en dat zij deze voorkeuren niet hoeven te veranderen. Vandaar dat voorkeur-behaven plan- en roostercoördinatiemechanismen nodig zijn, die toestaan dat agenten zelf hun voorkeuren specificeren en garanderen dat er een dispatching bestaat die voldoet aan alle gedefinieerde voorkeuren. Daarom zijn alleen a-priori coördinatie-via-ontwerp benaderingen acceptabel.

Door het opleggen van extra restricties aan het taaknetwerk, een zogenaamde coördinatie set, kunnen agenten worden gecoördineerd voor het onafhankelijke bouwen van plannen en roosters. We gebruiken planontkoppeling en roosterontkoppeling om, respectievelijk, de voorkeur-behandende coördinatiemechanismen voor de plannings- en roosterings fase mee aan te duiden.

We vonden een gat in de bestaande literatuur met betrekking tot coördinatiemechanismen voor zelfzuchtige planners en roosteraars. Enerzijds kon planontkoppeling nog niet worden toegepast wanneer andere restricties dan volgorde restricties gebruikt worden (bijvoorbeeld, synchronisatie restricties en tijdvensters). Rooster ontkoppeling, anderzijds, kon nog niet gebruikt worden voor een minder beperkte soort van roosteren dan dispatching (d.w.z., dispatch behoudend in plaats van voorkeur behoudend). Voor het overbruggen van deze kloof hebben we de volgende onderzoeksvragen geformuleerd.

Onderzoeksvraag 1: Is het mogelijk de toepasbaarheid van planontkoppeling uit te breiden naar expressievere raamwerken waarin andere temporele restricties toegestaan zijn, naast de volgorde restricties?

Het uitbreiden van de toepasbaarheid van planontkoppeling naar rijkere raamwerken met temporele restricties vereist dat we opnieuw de complexiteit van haar bijbehorende problemen bestuderen.

Onderzoeksvraag 2: Wat zijn de gevolgen voor planontkoppeling in termen van de complexiteit van het toevoegen van verschillende soorten temporele restricties aan het raamwerk?

Een taaknetwerk met kwalitatief-temporele restricties (bijvoorbeeld, volgorde restricties) kan worden ingebed in een taaknetwerk met kwantitatief-temporele restricties (bijvoorbeeld een eenvoudig temporeel netwerk). Door deze relatie kunnen we plan- en roosterontkoppeling vergelijken middels de volgende vragen.

Onderzoeksvraag 3: Wat is het verschil tussen plan- en roosterontkoppeling (in een kwantitatief raamwerk)?
Als er een verschil is tussen deze twee voorkeurs-behoudende benaderingen, zijn we geïnteresseerd in het verschil in de computationele complexiteit.

**Onderzoeksvraag 4:** Hoe verhoudt de complexiteit van planontkoppeling zich tot die van roosterontkoppeling (in kwantitatief raamwerk)?

Naast bovenstaande theoretisch gemotiveerde vragen, zijn wij geïnteresseerd in de praktische gevolgen van het gebruik van ontkoppeling. Het moet immers in het belang van de agent zijn om ontkoppeling toe te passen (d.w.z., de kosten mogen niet hoger zijn dan de baten), als de agenten zelfzuchtig zijn. Hier maken we een onderscheid tussen de gevolgen van het gebruik van planontkoppeling voor alle agenten gezamenlijk en het effect op een agent individueel.

**Onderzoeksvraag 5:** Wat is het effect van het gebruik van planontkoppeling op de plan-uitvoerings kosten en de plan-constructie tijden voor alle agenten gezamenlijk?

**Onderzoeksvraag 6:** Wat is het effect van het gebruik van planontkoppeling op de plan-uitvoerings kosten en de plan-constructie tijden voor de agenten individueel?

In hoofdstuk 3 bestuderen we planontkoppeling van een set taken beperkt door kwalitatief-temporele restricties, een zogenaamd kwalitatief taaknetwerk. We definiëren deze kwalitatieve taaknetwerken om een set van toegewezen primitieve taken te kunnen representeren, die worden beperkt door volgorde en synchronisatie restricties.

In deze kwalitatieve taaknetwerken illustreren we het plan-coördinatieprobleem dat ontstaat als alle agenten lokaal-consistente plannen mogen construeren. Twee varianten van het plan-ontkoppelings probleem zijn gedefinieerd voor de bouw van hetzij een subset-minimale ofwel een kardinaal-minimale coördinatie set. We bewijzen dat de beslissings problemen voor beide varianten in het algemeen ondoenlijk zijn. We zien dat de complexiteit van de problemen niet toeneemt als, naast volgorde restricties, ook synchronisatie restricties worden toegestaan. We bestuderen ook de complexiteit van subklassen waar of het aantal agenten of het aantal taken per agent beperkt is, en we bewijzen dat deze subklassen ondoenlijk zijn. Tenslotte formuleren we een heuristiek die in polynomiale tijd een planontkoppeling kan construeren, en geven een bewijs van correctheid.

In hoofdstuk 4 breiden we de kwalitatieve taaknetwerken van hoofdstuk 3 uit zodat kwantitatief-temporele restricties kunnen worden vermeld (zoals tijdsvensters en taakduren). In deze eenvoudige temporele taaknetwerken (STTNs) trekken we twee coördinatie problemen op die wij het plancoördinatieprobleem en het roostercoördinatieprobleem noemen. Het plan-coördinatieprobleem doet zich voor als voor lokaal consistent plannen samen mischien geen dispatching bestaat (bijv. als de volgorde restricties een cyclische afhankelijkheid bevat). Het roostercoördinatieprobleem doet zich voor als lokaal consistente roosters mogelijk niet
compatibel met elkaar zijn. Wij bestuderen plan- en roosterontkoppeling voor het oplossen van deze coördinatieproblemen.

Ten eerste breiden we de definities uit van de plan-ontkoppelings problemen en bestuderen hun complexiteit. We laten zien dat het plan-ontkoppeling erkenning probleem in een STTN en de kardinaal-minimal plan-ontkoppeling probleem beide ondoenlijk. De computacionele complexiteit van de plan-ontkoppelings problemen wordt niet groter in vergelijking met hun tegenhangers voor kwalitatieve taaknetwerken, ondanks de toegevoegde expressiviteit in het taaknetwerk.

Ten tweede breiden we de coördinatie aanpak uit voor de planning (d.w.z., om het bestaan van een gezamenlijke dispatching te garanderen) naar multi-agent taakgericht roosteren. Daarom hebben we roostercoördinatie gedefinieerd dat agenten toestaat om zelfstandig lokale roosters te construeren (d.w.z., kwantitatief-temporele restricties toe te voegen) met de garantie dat er geen conflict ontstaat tussen de agenten (d.w.z., dat er een gezamenlijke dispatching bestaat). Daarnaast hebben we dispatchcoördinatie gedefinieerd dat agenten toestaat om zelfstandig lokale dispatchings te construeren met de garantie dat hun lokale dispatchings samen een gezamenlijke dispatching vormen (d.w.z., vormt een oplossing voor het probleem). We ontdekten dat een STTN rooster gecoördineerd is als het dispatching gecoördineerd is, en vice versa. Daarom kan temporele ontkoppeling (Hunsberger, 2002) worden toegepast (d.w.z., een coördinatiemechanisme dat garandeert dat het samenvoegen van de lokale dispatchings een gezamenlijke dispatching vormt) zodat de agenten dispatchings kunnen construeren voor het uitvoeren van de toegewezen sets van taken.

In hoofdstuk 5 bestuderen we de praktische gevolgen als zelfzichtige agenten planontkoppeling gebruiken. In plaats van te kijken naar (de constructie van) de coördinatie set, richten we ons op het effect van planontkoppeling op de planuitvoerings kosten en de plan-constructie tijden. Meer specifiek bestuderen we deze effecten zowel vanuit een systeemperspectief als een agentperspectief om de potentiële kosten en baten te schatten als agenten hun autonomie behouden door met de andere agenten te coördineren.

Vanuit een systeemperspectief, vonden we dat de prijs van autonomie (d.w.z., de verhouding tussen de plan-uitvoerings kosten met en zonder planontkoppeling) gemiddeld dichtbij 1 te zijn en vaak zelfs kleiner (d.w.z., de aanpak met planontkoppeling resulteerde in goedkopere plannen) voor zowel de sequentiële (d.w.z., totale kosten) en parallelle planning (d.w.z. doorlooptijd), onafhankelijk van de grootte van de probleeminstantie. Voor de versnelling (d.w.z., de verhouding van het planconstructie tijd zonder en met planontkoppeling), vonden we waarden groter dan 1 die kunnen convergeren naar plannerspecifieke constanten of exponentieel kunnen groeien, bij het vergroten van de probleeminstantie.

Vanuit een agentperspectief, vonden we voor prijs van coördinatie (d.w.z., de verhouding tussen de plan-uitvoerings kosten met en zonder een coördinatiemechanisme) waarden zowel groter dan als kleiner dan 1 voor (heuristische) sequentiële planners, en groter dan of gelijk aan 1 voor (heuristische) parallel planning. Deze resultaten lijken erop te wijzen dat—vanuit een agentperspectief—
de prijs voor de coördinatie van parallelle planning hoger is dan voor sequentiële planning. Voor de versnelling vonden we waarden groter dan 1 voor heuristische sequentiële planners, en waarden zowel groter dan als kleiner dan 1 voor optimale planners.

In hoofdstuk 6 geven we de belangrijkste conclusies van dit proefschrift. Onderzoeksvraag 1 kan positief beantwoord worden, omdat we de toepasbaarheid van planontkoppeling hebben uitgebreid naar taaknetwerken met rijkere kwalitatief-temporele restricties (o.a. synchronisatie restricties) en kwantitatief-temporele restricties (o.a. tijdsvensters en taakduren). Dit resultaat toont de flexibiliteit van het raamwerk met betrekking tot de uitbreidbaarheid met nieuwe soorten restricties. Het antwoord op onderzoeksvraag 2 is dat de computationele complexiteit van planontkoppeling niet verandert voor het uitbreiden van het raamwerk met de beschouwde temporele restricties. Met betrekking tot onderzoeksvraag 3 kunnen we zeggen dat plan- en roosterontkoppeling verschillen in de soorten van coördinatie problemen die ze oplossen, omdat de potentiële conflicten verschillen bij het toevoegen van plannings of roosterings voorkeuren (d.w.z., kwalitatief-temporele of kwantitatief-temporele restricties). Omdat roosterontkoppeling meer restricties optingt dan planontkoppeling wordt plan van coördinatie geïmpliceerd door roostercoördinatie, terwijl het omgekeerde geen stand houdt. Voor onderzoeksvraag 4, vonden we dat de complexiteit van plan- en roosterontkoppeling verschilt (d.w.z., ondoenlijk versus doenlijk). Het betrekking tot het effect van het gebruik van planontkoppeling vanuit een systeemperspectief (onderzoeksvraag 5) vonden we dat de prijs van autonomie gemiddeld in de buurt en vaak zelfs kleiner dan 1 is (d.w.z., de gedecentraliseerde aanpak resulteerde in goedkopere plannen) voor zowel sequentiële planning (d.w.z., totale kosten) als parallel planning (d.w.z. doorlooptijd), onafhankelijk van de grootte van de probleeminstantie. Voor de versnelling vonden we waarden groter dan 1 die kunnen convergeren naar planner-specifieke constanten of exponentieel kunnen toenemen, bij het vergroten van de probleeminstantie. Met betrekking tot het effect van het gebruik van planontkoppeling vanuit een agentperspectief (onderzoeksvraag 6), vonden we dat de prijs van coördinatie meestal groter dan of gelijk aan 1 is voor (heuristische) sequentiële planning en groter dan of gelijk aan 1 voor (heuristische) parallel planning. Voor de versnelling vonden we waarden groter dan 1 voor heuristische sequentiële planners, en waarden zowel groter dan als kleiner dan 1 voor optimale planners.

In dit proefschrift hebben we met succes gebruik gemaakt van een aanpak in twee fasen met autonoom plannen, gevolgd door autonoom roosteren, om zelfzuchtige agenten in staat te stellen om onafhankelijk voorkeuren voor plannen en roosters op de uitvoering van de taken te specifiseren. We associëren kwalitatief-temporele restricties met de plannings voorkeuren (of strategische beslissingen), en associëren kwantitatief-temporele restricties met de roosterings voorkeuren (of tactische beslissingen) van de agenten. De bestudeerde a-priori plan- en roosterc-oördinatiemechanismen d.m.v. ontkoppeling zijn voorkeurs-behoudend geworden in de zin dat (i) de opgelegde voorkeursrestricties achteraf niet herzien hoeven te worden en (ii) de dispatching alle voorkeursrestricties honoreert.
Samenvatting
Jan Renze Steenhuisen was born on the 25th of January in 1979 in Hoogeveen, The Netherlands. After obtaining his VWO diploma at the Christelijk Gymnasium Utrecht in Utrecht, he started studying Computer Science at the Delft University of Technology. In 2004, he received his B.Sc. degree in computer science and his M.Sc./ir degree in computer science in the following year. After his M.Sc. project with the Parallel and Distributed Systems group, he moved on to become a PhD student as a member of the Algorithmics group of Prof. dr. Cees Witteveen, Delft University of Technology.

Renze’s research interests are with the coordination of multi-agent systems, with a focus on multi-agent planning systems, and with computer chess, with foci on deep-search behaviour and automatic openingbook construction.

Between 2005 and 2010, Renze was involved in various teaching activities, including the supervision of three M.Sc. students. He was involved in the research community as conference/workshop program committee member, and peer reviewer. Since 2010, he works as a software engineer at Quinity B.V., in Utrecht.
SIKS dissertations series

1998

1 Johan van den Akker (CWI) Degas - An Active, Temporal Database of Autonomous Objects
2 Floris Wiesman (UM) Information Retrieval by Graphically Browsing Meta-Information
3 Ans Steuten (TUD) A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective
4 Dennis Breuker (UM) Memory versus Search in Games
5 Eduard W. Oskamp (RUL) Computerondersteuning bij Straftoemeting

1999

1 Mark Sloof (VU) Physiology of Quality Change Modelling; Automated Modelling of Quality Change of Agricultural Products
2 Rob Potharst (EUR) Classification using Decision Trees and Neural Nets
3 Don Beal (UM) The Nature of Minimax Search
4 Jacques Penders (UM) The Practical Art of Moving Physical Objects
5 Aldo de Moor (KUB) Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems
6 Nick J.E. Wijngaards (VU) Re-Design of Compositional Systems
7 David Spelt (UT) Verification Support for Object Database Design
8 Jacques H.J. Lenting (UM) Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation

2000

1 Frank Niessink (VU) Perspectives on Improving Software Maintenance
2 Koen Holtman (TU/e) Prototyping of CMS Storage Management
3 Carolien M.T. Metselaar (UvA) Sociaal-organisatorische Gevolgen van Kennistechnologie; een Procesbenadering en Actorperspectief

1 Abbreviations: SIKS - Dutch Research School for Information and Knowledge Systems; CWI - Centrum Wiskunde & Informatica, Amsterdam; EUR - Erasmus Universiteit, Rotterdam; KUB - Katholieke Universiteit Brabant, Tilburg; KUN - Katholieke Universiteit Nijmegen; OU - Open Universiteit; RUL - Rijksuniversiteit Leiden; RUN - Radboud Universiteit Nijmegen; TUD - Technische Universiteit Delft; TU/e - Technische Universiteit Eindhoven; UL - Universiteit Leiden; UM - Universiteit Maastricht; UT - Universiteit Twente, Enschede; UU - Universiteit Utrecht; UvA - Universiteit van Amsterdam; UvT - Universiteit van Tilburg; VU - Vrije Universiteit, Amsterdam.
4 Geert de Haan (VU) ETAG, A Formal Model of Competence Knowledge for User Interface Design
5 Ruud van der Pol (UM) Knowledge-Based Query Formulation in Information Retrieval
6 Rogier van Eijk (UU) Programming Languages for Agent Communication
7 Niels Peek (UU) Decision-Theoretic Planning of Clinical Patient Management
8 Veerle Coupé (EUR) Sensitivity Analysis of Decision-Theoretic Networks
9 Florian Waas (CWI) Principles of Probabilistic Query Optimization
10 Niels Nes (CWI) Image Database Management System Design Considerations, Algorithms and Architecture
11 Jonas Karlsson (CWI) Scalable Distributed Data Structures for Database Management

2001
1 Silja Renooij (UU) Qualitative Approaches to Quantifying Probabilistic Networks
2 Koen Hindriks (UU) Agent Programming Languages: Programming with Mental Models
3 Maarten van Someren (UvA) Learning as Problem Solving
4 Evgueni Smirnov (UM) Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets
5 Jacco van Ossenbruggen (VU) Processing Structured Hypermedia: A Matter of Style
6 Martijn van Welie (VU) Task-Based User Interface Design
7 Bastiaan Schonhage (VU) Diva: Architectural Perspectives on Information Visualization
8 Pascal van Eck (VU) A Compositional Semantic Structure for Multi-Agent Systems Dynamics
9 Pieter Jan ‘t Hoen (RUL) Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes
10 Maarten Sierhuis (UvA) Modeling and Simulating Work Practice BRAHMS: a Multiagent Modeling and Simulation Language for Work Practice Analysis and Design
11 Tom M. van Engers (VU) Knowledge Management: The Role of Mental Models in Business Systems Design

2002
1 Nico Lassing (VU) Architecture-Level Modifiability Analysis
2 Roelof van Zwol (UT) Modelling and Searching Web-based Document Collections
3 Henk Ernst Blok (UT) Database Optimization Aspects for Information Retrieval
4 Juan Roberto Castelo Valdueza (UU) The Discrete Acyclic Digraph Markov Model in Data Mining
5 Radu Serban (VU) The Private Cyberspace Modeling Electronic Environments Inhabited by Privacy-Concerned Agents
6 Laurens Mommers (UL) Applied Legal Epistemology; Building a Knowledge-based Ontology of the Legal Domain
7 Peter Boncz (CWI) Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications
8 Jaap Gordijn (VU) Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas
9 Willem-Jan van den Heuvel (KUB) Integrating Modern Business Applications with Objectified Legacy Systems
10 Brian Sheppard (UM) *Towards Perfect Play of Scrabble*
11 Wouter C.A. Wijngaards (VU) *Agent Based Modelling of Dynamics: Biological and Organisational Applications*
12 Albrecht Schmidt (UvA) *Processing XML in Database Systems*
13 Hongjing Wu (TU/e) *A Reference Architecture for Adaptive Hypermedia Applications*
14 Wieke de Vries (UU) *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*
15 Rik Eshuis (UT) *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*
16 Pieter van Langen (VU) *The Anatomy of Design: Foundations, Models and Applications*
17 Stefan Manegold (UvA) *Understanding, Modeling, and Improving Main-Memory Database Performance*

2003
1 Heiner Stuckenschmidt (VU) *Ontology-Based Information Sharing in Weakly Structured Environments*
2 Jan Broersen (VU) *Modal Action Logics for Reasoning About Reactive Systems*
3 Martijn Schuemie (TUD) *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*
4 Milan Petkovic (UT) *Content-Based Video Retrieval Supported by Database Technology*
5 Jos Lehmann (UvA) *Causation in Artificial Intelligence and Law – A Modelling Approach*
6 Boris van Schooten (UT) *Development and Specification of Virtual Environments*
7 Machiel Jansen (UvA) *Formal Explorations of Knowledge Intensive Tasks*
8 Yong-Ping Ran (UM) *Repair-Based Scheduling*
9 Rens Kortmann (UM) *The Resolution of Visually Guided Behaviour*
10 Andreas Lincke (UT) *Electronic Business Negotiation: Some Experimental Studies on the Interaction between Medium, Innovation Context and Cult*
11 Simon Keizer (UT) *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*
12 Roeland Ordeman (UT) *Dutch Speech Recognition in Multimedia Information Retrieval*
13 Jeroen Donkers (UM) *Nosce Hostem – Searching with Opponent Models*
14 Stijn Hoppenbrouwers (KUN) *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*
15 Mathijs de Weerdt (TUD) *Plan Merging in Multi-Agent Systems*
16 Menzo Windhouwer (CWI) *Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouse*
17 David Jansen (UT) *Extensions of Statecharts with Probability, Time, and Stochastic Timing*
18 Levente Kocsis (UM) *Learning Search Decisions*

2004
1 Virginia Dignum (UU) *A Model for Organizational Interaction: Based on Agents, Founded in Logic*
2 Lai Xu (UvT) *Monitoring Multi-party Contracts for E-business*
3 Perry Groot (VU) A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving
4 Chris van Aart (UvA) Organizational Principles for Multi-Agent Architectures
5 Viara Popova (EUR) Knowledge Discovery and Monotonicity
6 Bart-Jan Hommes (TUD) The Evaluation of Business Process Modeling Techniques
7 Elise Boltjes (UM) VoorbeeldOnderwijs; Voorbeeldgestuurd Onderwijs, een Opstap naar Abstract Denken, vooral voor Meisjes
8 Joop Verbeek (UM) Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale Politiële Gegevensuitwisseling en Digitale Expertise
9 Martin Caminada (VU) For the Sake of the Argument; Explorations into Argument-based Reasoning
10 Suzanne Kabel (UvA) Knowledge-rich Indexing of Learning-objects
11 Michel Klein (VU) Change Management for Distributed Ontologies
12 The Duy Bui (UT) Creating Emotions and Facial Expressions for Embodied Agents
13 Wojciech Jamroga (UT) Using Multiple Models of Reality: On Agents who Know how to Play
14 Paul Harrenstein (UU) Logic in Conflict. Logical Explorations in Strategic Equilibrium
15 Arno Knobbe (UU) Multi-Relational Data Mining
16 Federico Divina (VU) Hybrid Genetic Relational Search for Inductive Learning
17 Mark Winands (UM) Informed Search in Complex Games
18 Vania Bessa Machado (UvA) Supporting the Construction of Qualitative Knowledge Models
19 Thijs Westerveld (UT) Using generative probabilistic models for multimedia retrieval
20 Madelon Evers (Nyenrode) Learning from Design: facilitating multidisciplinary design teams

2005

1 Floor Verdenius (UvA) Methodological Aspects of Designing Induction-Based Applications
2 Erik van der Werf (UM) AI techniques for the game of Go
3 Franc Grootjen (RUN) A Pragmatic Approach to the Conceptualisation of Language
4 Nirvana Meratnia (UT) Towards Database Support for Moving Object data
5 Gabriel Infante-Lopez (UvA) Two-Level Probabilistic Grammars for Natural Language Parsing
6 Pieter Spronck (UM) Adaptive Game AI
7 Flavius Frasincar (TU/e) Hypermedia Presentation Generation for Semantic Web Information Systems
8 Richard Vdovjak (TU/e) A Model-driven Approach for Building Distributed Ontology-based Web Applications
9 Jeen Broekstra (VU) Storage, Querying and Inferencing for Semantic Web Languages
10 Anders Bouwer (UvA) Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments
11 Elth Ogston (VU) Agent Based Matchmaking and Clustering - A Decentralized Approach to Search
12 Csaba Boer (EUR) Distributed Simulation in Industry
13 Fred Hamburg (UL) Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen
14 Borys Omelayenko (VU) Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics
15 Tibor Bosse (VU) Analysis of the Dynamics of Cognitive Processes
16 Joris Graaumans (UU) Usability of XML Query Languages
17 Boris Shishkov (TUD) Software Specification Based on Re-usable Business Components
18 Danielle Sent (UU) Test-selection strategies for probabilistic networks
19 Michel van Dartel (UM) Situated Representation
20 Cristina Coteanu (UL) Cyber Consumer Law, State of the Art and Perspectives
21 Wijnand Derks (UT) Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

2006
1 Samuil Angelov (TU/e) Foundations of B2B Electronic Contracting
2 Cristina Chisalita (VU) Contextual issues in the design and use of information technology in organizations
3 Noor Christoph (UvA) The role of metacognitive skills in learning to solve problems
4 Marta Sabou (VU) Building Web Service Ontologies
5 Cees Pierik (UU) Validation Techniques for Object-Oriented Proof Outlines
6 Ziv Baida (VU) Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling
7 Marko Smiljanic (UT) XML schema matching – balancing efficiency and effectiveness by means of clustering
8 Eelco Herder (UT) Forward, Back and Home Again - Analyzing User Behavior on the Web
9 Mohamed Wahdan (UM) Automatic Formulation of the Auditor’s Opinion
10 Ronny Siebes (VU) Semantic Routing in Peer-to-Peer Systems
11 Joeri van Ruth (UT) Flattening Queries over Nested Data Types
12 Bert Bongers (VU) Interactivation - Towards an e-cology of people, our technological environment, and the arts
13 Henk-Jan Lebbink (UU) Dialogue and Decision Games for Information Exchanging Agents
14 Johan Hoorn (VU) Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change
15 Rainer Malik (UU) CONAN: Text Mining in the Biomedical Domain
16 Carsten Riggelsen (UU) Approximation Methods for Efficient Learning of Bayesian Networks
17 Stacey Nagata (UU) User Assistance for Multitasking with Interruptions on a Mobile Device
18 Valenti Zhizhkun (UvA) Graph transformation for Natural Language Processing
19 Birna van Riemsdijk (UU) Cognitive Agent Programming: A Semantic Approach
20 Marina Velikova (UvT) Monotone models for prediction in data mining
21 Bas van Gils (RUN) Aptness on the Web
22 Paul de Vrieze (RUN) Fundaments of Adaptive Personalisation
23 Ion Juvina (UU) Development of Cognitive Model for Navigating on the Web
24 Laura Hollink (VU) Semantic Annotation for Retrieval of Visual Resources
25 Madalina Drugan (UU) Conditional log-likelihood MDL and Evolutionary MCMC
26 Vojkan Mihajlovic (UT) Score Region Algebra: A Flexible Framework for Structured Information Retrieval
27 Stefano Bocconi (CWI) Vox Populi: generating video documentaries from semantically annotated media repositories
28 Borkur Sigurbjornsson (UvA) Focused Information Access using XML Element Retrieval

2007
1 Kees Leune (UvT) Access Control and Service-Oriented Architectures
2 Wouter Teepe (RUG) Reconciling Information Exchange and Confidentiality: A Formal Approach
3 Peter Mika (VU) Social Networks and the Semantic Web
5 Bart Schermer (UL) Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance
6 Gilad Mishne (UvA) Applied Text Analytics for Blogs
7 Natasa Jovanovic’ (UT) To Whom It May Concern - Addressee Identification in Face-to-Face Meetings
8 Mark Hoogendoorn (VU) Modeling of Change in Multi-Agent Organizations
9 David Mobach (VU) Agent-Based Mediated Service Negotiation
10 Huib Aldewereld (UU) Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols
11 Natalia Stash (TU/e) Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System
12 Marcel van Gerven (RUN) Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty
13 Rutger Rienks (UT) Meetings in Smart Environments; Implications of Progressing Technology
14 Nick Bergboer (UM) Context-Based Image Analysis
15 Joyca Lacroix (UM) NIM: a Situated Computational Memory Model
16 Davide Grossi (UU) Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems
17 Theodore Charitos (UU) Reasoning with Dynamic Networks in Practice
18 Bart Orriens (UvT) On the development and management of adaptive business collaborations
19 David Levy (UM) Intimate relationships with artificial partners
20 Slinger Jansen (UU) Customer Configuration Updating in a Software Supply Network
21 Kariinne Vermaas (UU) Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005
22 Zlatko Zlatev (UT) Goal-oriented design of value and process models from patterns
23 Peter Barna (TU/e) Specification of Application Logic in Web Information Systems
24 Georgina Ramirez Camps (CWI) Structural Features in XML Retrieval
2008

1 Katalin Boer-Sorbán (EUR) Agent-Based Simulation of Financial Markets: A modular, continuous-time approach
3 Vera Hollink (UvA) Optimizing hierarchical menus: a usage-based approach
4 Ander de Keijzer (UT) Management of Uncertain Data - towards unattended integration
5 Bela Mutschler (UT) Modeling and simulating causal dependencies on process-aware information systems from a cost perspective
6 Arjen Hommersom (RUN) On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective
7 Peter van Rosmalen (OU) Supporting the tutor in the design and support of adaptive e-learning
8 Janneke Bolt (UU) Bayesian Networks: Aspects of Approximate Inference
9 Christof van Nimwegen (UU) The paradox of the guided user: assistance can be counter-effective
10 Wouter Bosma (UT) Discourse oriented Summarization
11 Vera Kartseva (VU) Designing Controls for Network Organizations: a Value-Based Approach
12 Jozsef Farkas (RUN) A Semiotically oriented Cognitive Model of Knowledge Representation
13 Caterina Carraciolo (UvA) Topic Driven Access to Scientific Handbooks
14 Arthur van Bunningen (UT) Context-Aware Querying; Better Answers with Less Effort
16 Henriette van Vuurt (VU) Embodied Agents from a User’s Perspective
17 Martin Op’t Land (TUD) Applying Architecture and Ontology to the Splitting and Allying of Enterprises
18 Guido de Croon (UM) Adaptive Active Vision
19 Henning Rode (UT) From document to entity retrieval: improving precision and performance of focused text search
20 Rex Arendsen (UvA) Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met een overheid op de administratieve lasten van bedrijven
21 Krisztian Balog (UvA) People search in the enterprise
22 Henk Koning (UU) Communication of IT-architecture
23 Stefan Visscher (UU) Bayesian network models for the management of ventilator-associated pneumonia
24 Zharko Aleksovski (VU) Using background knowledge in ontology matching
25 Geert Jonker (UU) Efficient and Equitable exchange in air traffic management plan repair using spender-signed currency
26 Marijn Huijbregts (UT) Segmentation, diarization and speech transcription: surprise data unraveled
27 Hubert Vogten (OU) Design and implementation strategies for IMS learning design
28 Ildiko Flesh (RUN) On the use of independence relations in Bayesian networks
29 Dennis Reidsma (UT) Annotations and subjective machines- Of annotators, embodied agents, users, and other humans
30 Wouter van Atteveldt (VU) Semantic network analysis: techniques for extracting, representing and querying media content
31 Loes Braun (UM) Pro-active medical information retrieval
32 Trung B. Hui (UT) Toward affective dialogue management using partially observable markov decision processes
33 Frank Terpstra (UvA) Scientific workflow design; theoretical and practical issues
34 Jeroen de Knijf (UU) Studies in Frequent Tree Mining
35 Benjamin Torben-Nielsen (UvT) Dendritic morphology: function shapes structure

2009

1 Rasa Jurgelenaite (RUN) Symmetric Causal Independence Models
2 Willem Robert van Hage (VU) Evaluating Ontology-Alignment Techniques
3 Hans Stol (UvT) A Framework for Evidence-based Policy Making Using IT
4 Josephine Nabukenya (RUN) Improving the Quality of Organisational Policy Making using Collaboration Engineering
5 Sietse Overbeek (RUN) Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality
6 Muhammad Subianto (UU) Understanding Classification
7 Ronald Poppe (UT) Discriminative Vision-Based Recovery and Recognition of Human Motion
8 Volker Nannen (VU) Evolutionary Agent-Based Policy Analysis in Dynamic Environments
9 Benjamin Kanagwa (RUN) Design, Discovery and Construction of Service-oriented Systems
10 Jan Wielemaker (UvA) Logic programming for knowledge-intensive interactive applications
11 Alexander Boer (UvA) Legal Theory, Sources of Law & the Semantic Web
12 Peter Massuthe (TU/e, Humboldt-Universität zu Berlin) Operating Guidelines for Services
13 Steven de Jong (UM) Fairness in Multi-Agent Systems
14 Maksym Korotkiy (VU) From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)
15 Rinke Hoekstra (UvA) Ontology Representation - Design Patterns and Ontologies that Make Sense
16 Fritz Reul (UvT) New Architectures in Computer Chess
17 Laurens van der Maaten (UvT) Feature Extraction from Visual Data
18 Fabian Groffen (CWI) Armada, An Evolving Database System
19 Valentin Robu (CWI) Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets
20 Bob van der Vecht (UU) Adjustable Autonomy: Controlling Influences on Decision Making
21 Stijn Vanderlooy (UM) Ranking and Reliable Classification
22 Pavel Serdyukov (UT) Search For Expertise: Going beyond direct evidence
23 Peter Hofgesang (VU) Modelling Web Usage in a Changing Environment
24 Annerieke Heuvelink (VU) Cognitive Models for Training Simulations
25 Alex van Ballegooij (CWI) “RAM: Array Database Management through Relational Mapping”
26 Fernando Koch (UU) An Agent-Based Model for the Development of Intelligent Mobile Services
27 Christian Glahn (OU) Contextual Support of social Engagement and Reflection on the Web
28 Sander Evers (UT) Sensor Data Management with Probabilistic Models
29 Stanislav Pokraev (UT) Model-Driven Semantic Integration of Service-Oriented Applications
30 Marcin Zukowski (CWI) Balancing vectorized query execution with bandwidth-optimized storage
31 Sofiya Katrenko (UvA) A Closer Look at Learning Relations from Text
32 Rik Farenhorst and Remco de Boer (VU) Architectural Knowledge Management: Supporting Architects and Auditors
33 Khiet Truong (UT) How Does Real Affect Affect Affect Recognition In Speech?
34 Inge van de Weerd (UU) Advancing in Software Product Management: An Incremental Method Engineering Approach

2010
1 Matthijs van Leeuwen (UU) Patterns that Matter
2 Ingo Wassink (UT) Work flows in Life Science
4 Olga Kulyk (UT) Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments
5 Claudia Hauff (UT) Predicting the Effectiveness of Queries and Retrieval Systems
6 Sander Bakkes (UvT) Rapid Adaptation of Video Game AI
7 Wim Fikkert (UT) A Gesture interaction at a Distance
8 Krzysztof Siewicz (UL) Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments
9 Hugo Kielman (UL) Politie¨ele gegevensverwerking en Privacy, Naar een effectieve waarborging
10 Rebecca Ong (UL) Mobile Communication and Protection of Children
11 Adriaan Ter Mors (TUD) The world according to MARP: Multi-Agent Route Planning
12 Susan van den Braak (UU) Sensemaking software for crime analysis
13 Gianluigi Folino (RUN) High Performance Data Mining using Bio-inspired techniques
14 Sander van Splunter (VU) Automated Web Service Reconfiguration
15 Lianne Bodenstaff (UT) Managing Dependency Relations in Inter-Organizational Models
16 Sicco Verwer (TUD) Efficient Identification of Timed Automata, theory and practice
17 Spyros Kotoulas (VU) Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications
18 Charlotte Gerritsen (VU) Caught in the Act: Investigating Crime by Agent-Based Simulation
19 Henriette Cramer (UvA) People’s Responses to Autonomous and Adaptive Systems
20 Ivo Swartjes (UT) Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative
21 Harold van Heerde (UT) Privacy-aware data management by means of data degradation
22 Michiel Hildebrand (CWI) End-user Support for Access to Heterogeneous Linked Data
23 Bas Steunebrink (UU) *The Logical Structure of Emotions*
24 Dmytro Tykhonov () *Designing Generic and Efficient Negotiation Strategies*
25 Zulfiqar Ali Memon (VU) *Modelling Human-Awareness for Ambient Agents: A Human Min- dreading Perspective*
26 Ying Zhang (CWI) *XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines*
27 Marten Voulon (UL) *Automatisch contracteren*
28 Arne Koopman (UU) *Characteristic Relational Patterns*
29 Stratos Idreos (CWI) *Database Cracking: Towards Auto-tuning Database Kernels*
30 Marieke van Erp (UvT) *Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval*
31 Victor de Boer (UvA) *Ontology Enrichment from Heterogeneous Sources on the Web*
32 Marcel Hiel (UvT) *An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems*
33 Robin Aly (UT) *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval*
34 Teduh Dirgahayu (UT) *Interaction Design in Service Compositions*
35 Dolf Trieschnigg (UT) *Proof of Concept: Concept-based Biomedical Information Retrieval*
36 Jose Janssen (OU) *Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification*
37 Niels Lohmann (TU/e) *Correctness of services and their composition*
38 Dirk Fahland (TU/e) *From Scenarios to components*
39 Ghazanfar Farooq Siddiqui (VU) *Integrative modeling of emotions in virtual agents*
40 Mark van Assem (VU) *Converting and Integrating Vocabularies for the Semantic Web*
41 Guillaume Chaslot (UM) *Monte-Carlo Tree Search*
42 Sybren de Kinderen (UvT) *Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach*
43 Peter van Krauwenburg (UU) *A Computational Approach to Content-Based Retrieval of Folk Song Melodies*
44 Pieter Bellekens (TU/e) *An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain*
45 Vasilios Andrikopoulos (UvT) *A theory and model for the evolution of software services*
46 Vincent Pijpers (VU) *e3alignment: Exploring Inter-Organizational Business-ICT Alignment*
47 Chen Li (UT) *Mining Process Model Variants: Challenges, Techniques, Examples*
48 Milan Lovric (EUR) *Behavioral Finance and Agent-Based Artificial Markets*
49 Jahn-Takeshi Saito (UM) *Solving difficult game positions*
50 Bouke Huurnink (UvA) *Search in Audiovisual Broadcast Archives*
51 Alia Khairia Amin (CWI) *Understanding and supporting information seeking tasks in multiple sources*
52 Peter-Paul van Maanen (VU) *Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention*
53 Edgar Meij (UvA) *Combining Concepts and Language Models for Information Access*
2011
1 Botond Cseke (RUN) Variational Algorithms for Bayesian Inference in Latent Gaussian Models
2 Nick Timmemeier (UU) Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language
3 Jan Martijn van der Werf (TU/e) Compositional Design and Verification of Component-Based Information Systems
4 Hado van Hasselt (UU) Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms
5 Base van der Raadt (VU) Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.
6 Yiweng Wang (TU/e) Semantically-Enhanced Recommendations in Cultural Heritage
7 Yujia Cao (UT) Multimodal Information Presentation for High Load Human Computer Interaction
8 Nieske Vergunst (UU) BDI-based Generation of Robust Task-Oriented Dialogues
9 Tim de Jong (OU) Contextualised Mobile Media for Learning
10 Bart Bogaert (UvT) Cloud Content Contention
11 Dhaval Vyas (UT) Designing for Awareness: An Experience-focused HCI Perspective
12 Carmen Bratosin (TU/e) Grid Architecture for Distributed Process Mining
13 Xiaoyu Mao (UvT) Airport under Control. Multiagent Scheduling for Airport Ground Handling
14 Milan Lovric (EUR) Behavioral Finance and Agent-Based Artificial Markets
15 Marijn Koolen (UvA) The Meaning of Structure: the Value of Link Evidence for Information Retrieval
16 Maarten Schadd (UM) Selective Search in Games of Different Complexity
17 Jiyin He (UvA) Exploring Topic Structure: Coherence, Diversity and Relatedness
18 Mark Ponsen (UM) Strategic Decision-Making in complex games
19 Ellen Rusman (OU) The Mind’s Eye on Personal Profiles
20 Qing Gu (VU) Guiding service-oriented software engineering - A view-based approach
21 Linda Torlouw (TUD) Modularization and Specification of Service-Oriented Systems
22 Junte Zhang (UvA) System Evaluation of Archival Description and Access
23 Wouter Weerkamp (UvA) Finding People and their Utterances in Social Media
24 Herwin van Welbergen (UT) Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior
25 Syed Waqar ul Qounain Jaffry (VU) Analysis and Validation of Models for Trust Dynamics
26 Matthijs Aart Pontier (VU) Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots
27 Aniel Bhulai (VU) Dynamic website optimization through autonomous management of design patterns
28 Rianne Kaptein (UvA) Effective Focused Retrieval by Exploiting Query Context and Document Structure
29 Faisal Kamiran (TU/e) Discrimination-aware Classification
30 Egon van den Broek (UT) Affective Signal Processing (ASP): Unraveling the mystery of emotions
31 Ludo Waltman (EUR) Computational and Game-Theoretic Approaches for Modeling Bounded Rationality
32 Nees-Jan van Eck (EUR) Methodological Advances in Bibliometric Mapping of Science
33 Tom van der Weide (UU) Arguing to Motivate Decisions
34 Paolo Turrini (UU) Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations
35 Maaike Harbers (UU) Explaining Agent Behavior in Virtual Training
36 Erik van der Spek (UU) Experiments in serious game design: a cognitive approach
37 Adriana Burlutiu (RUN) Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference
38 Nyree Lemmens (UM) Bee-inspired Distributed Optimization
39 Joost Westra (UU) Organizing Adaptation using Agents in Serious Games
40 Viktor Clerc (VU) Architectural Knowledge Management in Global Software Development
41 Luan Ibraimi (UT) Cryptographically Enforced Distributed Data Access Control
42 Michal Sindlar (UU) Explaining Behavior through Mental State Attribution
43 Henk van der Schuur (UU) Process Improvement through Software Operation Knowledge
44 Boris Reuderink (UT) Robust Brain-Computer Interfaces
45 Herman Stehouwer (UvT) Statistical Language Models for Alternative Sequence Selection
46 Beibei Hu (TUD) Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work
47 Azizi Bin Ab Aziz (VU) Exploring Computational Models for Intelligent Support of Persons with Depression
48 Mark Ter Maat (UT) Response Selection and Turn-taking for a Sensitive Artificial Listening Agent
49 Andreea Niculescu (UT) Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

2012

1 Terry Kakeeto (UvT) Relationship Marketing for SMEs in Uganda
2 Muhammad Umair (VU) Adaptivity, emotion, and Rationality in Human and Ambient Agent Models
3 Adam Vanya (VU) Supporting Architecture Evolution by Mining Software Repositories
4 Jurriaan Souer (UU) Development of Content Management System-based Web Applications
5 Marijn Plomp (UU) Maturing Interorganisational Information Systems
6 Wolfgang Reinhardt (OU) Awareness Support for Knowledge Workers in Research Networks
7 Rianne van Lumba gen (VU) When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions
8 Gerben de Vries (UvA) Kernel Methods for Vessel Trajectories
9 Ricardo Neisse (UT) Trust and Privacy Management Support for Context-Aware Service Platforms
10 David Smits (TU/e) Towards a Generic Distributed Adaptive Hypermedia Environment
<table>
<thead>
<tr>
<th></th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>J.C.B. Rantham Prabhakara (TU/e)</td>
<td>Process Mining in the Large: Preprocessing, Discovery, and Diagnostics</td>
</tr>
<tr>
<td>12</td>
<td>Kees van der Sluijs (TU/e)</td>
<td>Model Driven Design and Data Integration in Semantic Web Information Systems</td>
</tr>
<tr>
<td>13</td>
<td>Suleman Shahid (UvT)</td>
<td>Fun and Face: Exploring non-verbal expressions of emotion during playful interactions</td>
</tr>
<tr>
<td>14</td>
<td>Evgeny Knutov (TU/e)</td>
<td>Generic Adaptation Framework for Unifying Adaptive Web-based Systems</td>
</tr>
<tr>
<td>15</td>
<td>Natalie van der Wal (VU)</td>
<td>Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes</td>
</tr>
<tr>
<td>16</td>
<td>Fiemke Both (VU)</td>
<td>Helping people by understanding them - Ambient Agents supporting task execution and depression treatment</td>
</tr>
<tr>
<td>17</td>
<td>Amal Elgammal (UvT)</td>
<td>Towards a Comprehensive Framework for Business Process Compliance</td>
</tr>
<tr>
<td>18</td>
<td>Eltjo Poort (VU)</td>
<td>Improving Solution Architecting Practices</td>
</tr>
<tr>
<td>19</td>
<td>Helen Schonenberg (TU/e)</td>
<td>What’s Next? Operational Support for Business Process Execution</td>
</tr>
<tr>
<td>20</td>
<td>Ali Bahramisharif (RUN)</td>
<td>Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing</td>
</tr>
<tr>
<td>21</td>
<td>Roberto Cornacchia (TUD)</td>
<td>Querying Sparse Matrices for Information Retrieval</td>
</tr>
<tr>
<td>22</td>
<td>Thijs Vis (UvT)</td>
<td>Intelligence, politie en veiligheidsdienst: verenigbare grootheden?</td>
</tr>
<tr>
<td>23</td>
<td>Christian Muehl (UT)</td>
<td>Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction</td>
</tr>
<tr>
<td>24</td>
<td>Laurens van der Werff (UT)</td>
<td>Evaluation of Noisy Transcripts for Spoken Document Retrieval</td>
</tr>
<tr>
<td>25</td>
<td>Silja Eckartz (UT)</td>
<td>Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application</td>
</tr>
<tr>
<td>26</td>
<td>Emile de Maat (UvA)</td>
<td>Making Sense of Legal Text</td>
</tr>
<tr>
<td>27</td>
<td>Hayrettin Gürkök (UT)</td>
<td>Mind the Sheep! User Experience Evaluation &amp; Brain-Computer Interface Games</td>
</tr>
<tr>
<td>28</td>
<td>Nancy Pascall (UvT)</td>
<td>Engendering Technology Empowering Women</td>
</tr>
<tr>
<td>29</td>
<td>Almer Tigelaar (UT)</td>
<td>Peer-to-Peer Information Retrieval</td>
</tr>
<tr>
<td>30</td>
<td>Alina Pommeranz (TUD)</td>
<td>Designing Human-Centered Systems for Reflective Decision Making</td>
</tr>
<tr>
<td>31</td>
<td>Emily Bagarukayo (RUN)</td>
<td>A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure</td>
</tr>
<tr>
<td>32</td>
<td>Wietske Visser (TUD)</td>
<td>Qualitative multi-criteria preference representation and reasoning</td>
</tr>
<tr>
<td>33</td>
<td>Rory Sie (OUN)</td>
<td>Coalitions in Cooperation Networks (COCOON)</td>
</tr>
<tr>
<td>34</td>
<td>Pavol Jancura (RUN)</td>
<td>Evolutionary analysis in PPI networks and applications</td>
</tr>
<tr>
<td>35</td>
<td>Evert Haasdijk (VU)</td>
<td>Never Too Old To Learn – On-line Evolution of Controllers in Swarm-and Modular Robotics</td>
</tr>
<tr>
<td>36</td>
<td>Denis Ssebugwawo (RUN)</td>
<td>Analysis and Evaluation of Collaborative Modeling Processes</td>
</tr>
<tr>
<td>37</td>
<td>Agnes Nakakawa (RUN)</td>
<td>A Collaboration Process for Enterprise Architecture Creation</td>
</tr>
<tr>
<td>38</td>
<td>Selmar Smit (VU)</td>
<td>Parameter Tuning and Scientific Testing in Evolutionary Algorithms</td>
</tr>
<tr>
<td>39</td>
<td>Hassan Fatemi (UT)</td>
<td>Risk-aware design of value and coordination networks</td>
</tr>
</tbody>
</table>
40 Agus Gunawan (UvT) Information Access for SMEs in Indonesia
41 Sebastian Kelle (OU) Game Design Patterns for Learning
42 Dominique Verpoorten (OU) Reflection Amplifiers in self-regulated Learning
43 (Withdrawn)
44 Anna Tordai (VU) On Combining Alignment Techniques
45 Benedikt Kratz (UvT) A Model and Language for Business-aware Transactions
46 Simon Carter (UvA) Exploration and Exploitation of Multilingual Data for Statistical Machine Translation
47 Manos Tsagkias (UvA) Mining Social Media: Tracking Content and Predicting Behavior
48 Jorn Bakker (TU/e) Handling Abrupt Changes in Evolving Time-series Data
49 Michael Kaisers (UM) Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions
50 Steven van Kervel (TUD) Ontology driven Enterprise Information Systems Engineering
51 Jeroen de Jong (TUD) Heuristics in Dynamic Scheduling: a practical framework with a case study in elevator dispatching

2013
1 Viorel Milea (EUR) News Analytics for Financial Decision Support
2 Erietta Liarou (CWI) MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing
3 Szymon Klarman (VU) Reasoning with Contexts in Description Logics
4 Chetan Yadati (TUD) Coordinating autonomous planning and scheduling
5 Dulce Pumareja (UT) Groupware Requirements Evolutions Patterns
6 Romulo Goncalves (CWI) The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience
7 Giel van Lankveld (UT) Quantifying Individual Player Differences
8 Robbert-Jan Merk (VU) Making enemies: cognitive modeling for opponent agents in fighter pilot simulators
9 Fabio Gori (RUN) Metagenomic Data Analysis: Computational Methods and Applications
10 Jeewanie Jayasinghe Arachchige (UvT) A Unified Modeling Framework for Service Design
11 Evangelos Pournaras (TUD) Multi-level Reconfigurable Self-organization in Overlay Services
12 Maryam Razavian (VU) Knowledge-driven Migration to Services
13 Mohammad Zafiri (UT) Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly
14 Jafar Tanha (UvA) Ensemble Approaches to Semi-Supervised Learning Learning
15 Daniel Hennes (UM) Multiagent Learning - Dynamic Games and Applications
16 Eric Kok (UU) Exploring the practical benefits of argumentation in multi-agent deliberation
17 Koen Kok (VU) The PowerMatcher: Smart Coordination for the Smart Electricity Grid
18 Jeroen Janssens (UvT) Outlier Selection and One-Class Classification
Nomenclature

\[ A \] set of agents
\[ A_i \] single agent
\[ T \] set of tasks
\[ \tau_i \] single task
\[ T \] set of time points
\[ t_i \] single time point
\[ < \] set of precedence constraints
\[ \equiv \] set of synchronisation constraints
\[ C \] set of temporal-distance constraints
\[ T \] set of time-window constraints
\[ \xi_{\text{inter}} \] set of inter-agent \( \xi \) constraints
\[ \langle \{ T_i \}_{i=1}^n, <, \equiv \rangle \] qualitative task network
\[ \langle \{ (T_i, C_i, I_i) \}_{i=1}^n, C_{\text{inter}} \rangle \] simple temporal task network
\[ \pi, C^\pi \] plan
\[ C^\sigma \] schedule
\[ \nu \] dispatching
\[ \Gamma \] coordination set
**Glossary**

**Definition D.1 (Plan).** A plan $\pi_i$ for a task network $\langle T_i, \prec_i, \equiv_i \rangle$ is a set of ordering constraints $\langle \pi_i^\prec, \pi_i^\equiv \rangle$ where $\pi_i^\prec$ and $\pi_i^\equiv$ are extensions of $\prec_i$ and $\equiv_i$, respectively.

**Definition D.2 (Dispatching).** A dispatching $\nu_i$ for a task network $\langle T_i, \prec_i, \equiv_i \rangle$ is a function $\nu_i : T_i \rightarrow Q$ such that:

1. $\forall t, t' \in T_i : t \prec_i t'$ implies $\nu_i(t) < \nu_i(t')$, and
2. $\forall t, t' \in T_i : t \equiv_i t'$ implies $\nu_i(t) = \nu_i(t')$.

**Definition D.3 (Plan Consistency).** A plan $\pi_i = \langle \pi_i^\prec, \pi_i^\equiv \rangle$ for a task network $\langle T_i, \prec_i, \equiv_i \rangle$ is called consistent if there exists a dispatching $\nu_i$ for the network $\langle T_i, \pi_i^\prec, \pi_i^\equiv \rangle$ that results after adding the planning constraints to the original task network.

**Definition D.4 (Joint Dispatching).** A joint dispatching $\nu = \{\nu_i\}_{i=1}^n$ for a task network $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ is a function $\nu : T \rightarrow Q$ such that:

1. $\nu_i$ is a dispatching for $\langle T_i, \prec_i, \equiv_i \rangle$,
2. $\forall t \in T : t \in T_i$ implies $\nu(t) = \nu_i(t)$,
3. $\forall t \in T_i, t' \in T_j : t < t'$ implies $\nu(t) < \nu(t')$, and
4. $\forall t \in T_i, t' \in T_j : t \equiv t'$ implies $\nu(t) = \nu(t')$.

**Definition D.5 (Joint Plan Consistency).** A joint plan $\{\{\pi_i^\prec\}_{i=1}^n, \{\pi_i^\equiv\}_{i=1}^n\}$ for task network $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ is called consistent if there exists a joint dispatching $\nu$ for $\langle \{T_i\}_{i=1}^n, \prec_{\text{inter}} \cup \{\pi_i^\prec\}_{i=1}^n, \equiv_{\text{inter}} \cup \{\pi_i^\equiv\}_{i=1}^n \rangle$.

**Definition D.6 (Plan Coordination).** A task network $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ is called plan coordinated when the joint plan $\pi = \{\pi_i\}_{i=1}^n$ is consistent, for every set $\{\pi_i\}_{i=1}^n$ of locally-consistent plans $\pi_i$.

**Definition D.7 (Plan Decoupling).** A plan decoupling of a task network $\langle \{T_i\}_{i=1}^n, \prec, \equiv \rangle$ is formed by adding a set of constraints $\Gamma = \Gamma^\prec \cup \Gamma^\equiv$ such that the resulting task network $\langle \{T_i\}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv \rangle$ is plan coordinated.
Definition D.8 (Dispatching). A dispatching $\nu$ for an STTN $\mathcal{S} = \langle T, \mathcal{C}, I \rangle$ is a function $\nu : T \rightarrow \mathbb{Q}$ such that:

1. $\forall t \in T : \nu(t) \in I(t)$, and
2. $\forall t, t' \in T : t' - t \leq \delta_{ij} \in \mathcal{C}$ implies $\nu(t') - \nu(t) \leq \delta_{ij}$.

Definition D.9 (Qualitative refinement). A qualitative refinement of a constraint $c_{ij} : t_j - t_i \leq \delta_{ij} \in \mathcal{C}$ in an STTN $\mathcal{S} = \langle T, \mathcal{C}, I \rangle$ is a tightening $c'_{ij} : t_j - t_i \leq \delta'_{ij}$, with $\delta'_{ij} \in \{-\epsilon, 0\}$ and $\delta'_{ij} < \delta_{ij}$.

Definition D.10 (Plan). A plan $\mathcal{C}^n$ for an STTN $\mathcal{S} = \langle T, \mathcal{C}, I \rangle$ is a set of constraints for which it holds that each constraint $c_{ij} \in \mathcal{C}$ is either equal to the original constraint $c_{ij} \in \mathcal{C}$, or $c'_{ij}$ is a qualitative refinement of the constraint $c_{ij} \in \mathcal{C}$.

Definition D.11 (Joint Dispatching). A joint dispatching $\nu = \{\nu_i\}_{i=1}^n$ for an STTN $\langle\{\langle T_i, \mathcal{C}_i, I_i \rangle\}_{i=1}^n, \mathcal{C}_{\text{inter}}\rangle$ is a function $\nu : T \rightarrow \mathbb{Q}$ such that:

1. $\nu_i$ is a dispatching for $\langle T_i, \mathcal{C}_i, I_i \rangle$,
2. $\forall t : t \in T_i$ implies $\nu(t) = \nu_i(t)$, and
3. $\forall t_i, t_j \in T_{j \neq i} : c_{ij} : t_j - t_i \leq \delta_{ij} \in \mathcal{C}_{\text{inter}}$ implies $\nu(t_j) - \nu(t_i) \leq \delta_{ij}$.

Definition D.12 (Joint Plan Consistency). A joint plan $\{\mathcal{C}^n_i\}_{i=1}^n$ for STTN $\langle\{\langle T_i, \mathcal{C}_i, I_i \rangle\}_{i=1}^n, \mathcal{C}_{\text{inter}}\rangle$ is called consistent if there exists a joint dispatching $\nu$ for $\langle\{\langle T_i, \mathcal{C}^n_i, I_i \rangle\}_{i=1}^n, \mathcal{C}_{\text{inter}}\rangle$.

Definition D.13 (Plan Coordination). An STTN $\mathcal{S} = \langle\{\langle T_i, \mathcal{C}_i, I_i \rangle\}_{i=1}^n, \mathcal{C}_{\text{inter}}\rangle$ is called plan coordinated when the joint plan $\mathcal{C}^n = \{\mathcal{C}^n_i\}_{i=1}^n$ is consistent, for every set $\{\mathcal{C}^n_i\}_{i=1}^n$ of locally-consistent plans $\mathcal{C}^n_i$.

Definition D.14 (Plan Decoupling). A plan decoupling for an STTN $\langle\{\langle T_i, \mathcal{C}_i, I_i \rangle\}_{i=1}^n, \mathcal{C}_{\text{inter}}\rangle$ is an STTN $\langle\{\langle T_i, \mathcal{C}'_i, I_i \rangle\}_{i=1}^n, \mathcal{C}_{\text{inter}}\rangle$, such that

1. each $\langle T_i, \mathcal{C}'_i, I_i \rangle$ is a qualitative-refined sub-task network of $\langle T_i, \mathcal{C}_i, I_i \rangle$, and
2. the STTN $\langle\{\langle T_i, \mathcal{C}'_i, I_i \rangle\}_{i=1}^n, \mathcal{C}_{\text{inter}}\rangle$ is plan coordinated.

Definition D.15 (Quantitative refinement). A quantitative refinement of a constraint in an STTN $\mathcal{S} = \langle T, \mathcal{C}, I \rangle$ is

- a tightening $c'_{ij} : t_j - t_i \leq \delta'_{ij}$ of a constraint $c_{ij} \in \mathcal{C}$, with $\delta'_{ij} < \delta_{ij}$, or
- a tightening $I'(t) = [lb'(t), ub'(t)]$ of a time window $I(t) = [lb(t), ub(t)]$, with $lb(t) < lb'(t)$ or $ub'(t) < ub(t)$. 
Definition D.16 (Schedule). A schedule $C^\sigma$ for an STTN $S = \langle T, C, I \rangle$ is a set of constraints for which it holds that each constraint $c'_{ij} \in C^\sigma$ is either equal to the original constraint $c_{ij} \in C$, or $c'_{ij}$ is a quantitative refinement of the constraint $c_{ij} \in C$.

Definition D.17 (Schedule Coordination). An STTN $S = \langle \{\langle T_i, C_i, I_i \rangle\}_{i=1}^n, C_{\text{inter}} \rangle$ is called schedule coordinated when, for every set of locally-consistent schedules $\{C_i^\sigma\}_{i=1}^n$, there exists a joint dispatching $\{\nu_i\}_{i=1}^n$ for $S^\sigma = \langle \{\langle T_i, C_i^\sigma, I_i \rangle\}_{i=1}^n, C_{\text{inter}} \rangle$.

Definition D.18 (Dispatch Coordination). An STTN $S = \langle \{\langle T_i, C_i, I_i \rangle\}_{i=1}^n, C_{\text{inter}} \rangle$ is called dispatch coordinated when every set of local dispatchings $\{\nu_i\}_{i=1}^n$ is a joint dispatching for $S$. 
List of Decision Problems

Plan-Decoupling Recognition Problem
INSTANCE: A task network \(\{T_i\}_{i=1}^n, \prec, \equiv\) and a set of constraints \(\Gamma = \Gamma^\prec \cup \Gamma^\equiv\).
QUESTION: Is \(\{T_i\}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv\) plan coordinated?

subset-Minimal Plan-Decoupling Problem
INSTANCE: A task network \(\{T_i\}_{i=1}^n, \prec, \equiv\), a coordination set \(\hat{\Gamma} = \hat{\Gamma}^\prec \cup \hat{\Gamma}^\equiv\), and a positive integer \(K\).
QUESTION: Does there exist a coordination set \(\Gamma = \Gamma^\prec \cup \Gamma^\equiv\), with \(\Gamma^\prec \subseteq \hat{\Gamma}^\prec\) and \(\Gamma^\equiv \subseteq \hat{\Gamma}^\equiv\), and \(|\Gamma| \leq K\), such that \(\{T_i\}_{i=1}^n, \prec \cup \Gamma^\prec, \equiv \cup \Gamma^\equiv\) is plan coordinated?

cardinal-Minimal Plan-Decoupling Problem
INSTANCE: A task network \(\{T_i\}_{i=1}^n, \prec, \equiv\) and a positive integer \(K\).
QUESTION: Does there exist a coordination set \(\Gamma = \bigcup_{i=1}^n \Gamma^\prec_i \cup \bigcup_{i=1}^n \Gamma^\equiv_i\) such that \(\{T_i\}_{i=1}^n, \prec \cup \bigcup_{i=1}^n \Gamma^\prec_i, \equiv \cup \bigcup_{i=1}^n \Gamma^\equiv_i\) is plan coordinated?

STTN Plan-Decoupling Recognition Problem
INSTANCE: An STTN \(\{\langle T_i, C_i, I_i \rangle\}_{i=1}^n, \Gamma_{\text{inter}}\) and a set of constraints \(\Gamma = \bigcup_{i=1}^n \Gamma^\prec_i \cup \bigcup_{i=1}^n \Gamma^\equiv_i\).
QUESTION: Does it hold that the qualitatively-refined STTN \(\{\langle T_i, C^\Gamma_i, I_i \rangle\}_{i=1}^n, \Gamma_{\text{inter}}\) is plan coordinated?

STTN cardinal-Minimal Plan-Decoupling Problem
INSTANCE: An STTN \(\{\langle T_i, C_i, I_i \rangle\}_{i=1}^n, \Gamma_{\text{inter}}\) and a positive integer \(K\).
QUESTION: Does there exist a coordination set \(\Gamma = \bigcup_{i=1}^n \Gamma^\prec_i \cup \bigcup_{i=1}^n \Gamma^\equiv_i\) with \(|\Gamma| \leq K\) such that the qualitatively-refined STTN \(\{\langle T_i, C^\Gamma_i, I_i \rangle\}_{i=1}^n, \Gamma_{\text{inter}}\) is plan coordinated?