Optimal Temporal Decoupling in Task Scheduling with Preferences

Leon Endhoven Tomas Klos Cees Witteveen

Department of Software Technology[1]
Delft University of Technology, Delft, The Netherlands

Abstract

Multi-agent planning and scheduling concerns finding a joint plan to achieve some set of common goals with several independent agents each aiming to find a plan or schedule for their part of the goals. To avoid conflicts in these individual plans or schedules *decoupling* is used. Such a decoupling entails adding local constraints for the agents such that they can schedule autonomously within those constraints, while they guarantee that a conflict-free global solution can be constructed from the individual agents' schedules. In this paper we investigate finding an 'optimal' decoupling, that maximizes the sum of the agents' preferences about their scheduling of tasks. We show using a Linear Programming (LP) approach that optimal decouplings can be found efficiently by exploiting the properties of a task scheduling instance.

1 Introduction

Planning and scheduling are very important but difficult-to-solve problems occurring in many domains. Intuitively, a planning problem consists of a current and a goal state, and a set of activities which are able to transform states into other states (see, e.g., [8]). A solution is a selection and ordering of activities, that transform the current state into the goal state, possibly via intermediate states. Here, we assume that the ordering of activities is already given, along with some quantitative temporal information about this ordering. The task that remains is that of scheduling the activities at time-points such that the temporal constraints are satisfied [3]. We focus on the very common situation of planning tasks that need multiple agents to be executed [7].

We will illustrate the issues at hand using the following running example.

Running Example. *Imagine that Alice, Bob and Chloe participate in a 3-hour science project which has to take place some day between 12.00h and 18.00h. Suppose they can divide the workload evenly between the three of them, resulting in 3 lab experiments of one hour each, to be done by Alice, Bob and Chloe, in that order, because Bob and Chloe each use the results of the previous experiment.*

Before she does the experiment, Alice wants to have lunch, which takes half an hour. After the experiment she wants to spend 2 hours on her homework. Bob wants to spend 2 hours on his homework before he does his part of the experiment. Once he has finished his experiment, he wants to prepare dinner, which takes 1 hour. Chloe aims to spend half an hour on her lunch, after which she wants to go cycling for 2 hours, but she needs at least half an hour time between lunch and cycling. Once she has returned from cycling, she wants to do her part of the experiment.

We would like Alice, Bob and Chloe to independently find a schedule for their activities such that when merged, these schedules constitute a global schedule satisfying all constraints.

The framework of the temporal constraint satisfaction problem (TCSP) can be used to model these situations [3]. In a TCSP, time variables represent events like the start or end of an activity, and sets of temporal intervals specify the allowed values for these variables as well as for differences between certain pairs of variables. Various questions may be asked of a given TCSP instance, such as whether it is consistent,

[1]The first author is a master student in Applied Mathematics at Delft University of Technology.

i.e., whether there exists an assignment of time values to the variables such that no constraint is violated, or whether a feasible schedule exists that satisfies some additional constraints. If for each variable and pair of variables there exists at most one interval constraining it, we speak of the Simple Temporal Problem (STP) [3], an instance of which is called a Simple Temporal Network (STN). Every vertex in an STN represents a time variable, for example the end of Bob's homework, while directed arcs represent the (precedence) relations between the tasks, and are labeled with the interval in which the difference between the two tasks has to lay.

When the time variables are distributed among a set of agents, such as in the example, we cannot in general let the agents schedule these time variables independently of one another, because there may be dependencies between the different agents' time variables. For example, if Alice wants to start her part of the experiment as late as possible, at 15h, followed by 2 hours of homework between 16h and 18h, while Bob wants to start his part of the experiment as early as possible, at 14h, after 2 hours of homework, then these schedules are consistent with the agents' preferences. However, the combination of these local agent-schedules violates the ordering constraint on the three parts of the experiment: Bob does his part of the experiment before Alice.

To prevent such problems, we need to *coordinate* the agents' plans and schedules [5]. Various approaches have been proposed for this in the literature. In *plan merging* [11], coordination is applied after plans have been constructed by the agents. Alternatively, *coordination during planning* [4] views planning and coordination as intertwined processes. Here we focus on the solution approach of *coordination by design* [7], inter-agent dependencies are removed before planning, by giving the agents extra local constraints, such that with the set of those constraints, they are free to plan autonomously, while conflicts are guaranteed not to occur upon merging the plans. Examples of this approach are temporal decoupling in Simple Temporal Networks [6] and plan decoupling in general task-based networks [1, 10]. The idea is that all inter-agent constraints are replaced by local constraints for the agents involved, where the interval specifying the difference between the different agents' variables is 'split up' between the two agents. In our example, we could say that Alice has to start her part of the experiment at 13.30h at the latest and Bob can start his part of the experiment not earlier than 14.30h. This way, no matter how the agents solve their local STN regarding these tasks, we are guaranteed that their local solutions, when combined, do not violate the inter-agent constraint.

Of course, there are (infinitely) many ways in which we can replace even a single inter-agent constraint with multiple local agent-constraints, so there are many alternative decouplings. The agents involved in the problem most likely have varying preferences for different decouplings, and would like to receive a decoupled STN in which their preferences have been taken into account. Finding such a decoupling that maximizes the sum of the agents' preferences is called the Optimal Decoupling Problem (ODP) [9]. While in STNs an arbitrary decoupling can be found efficiently [6], the ODP is NP-hard [9], though it is efficiently solvable if all agents have linear preference functions [9].

In this paper, we study this optimal decoupling problem in a subclass of STNs, of which our running example is an instance. In this subclass, unlike general STNs, preceding tasks have to be completely finished before a task can commence. After presenting preliminaries in section 2, we discuss our results for the temporal decoupling problem in this class of networks in section 3 and for the optimal decoupling problem in section 3. Section 4 concludes our paper.

2 Preliminaries

2.1 Task Scheduling

The example given in the introduction is an example of a multi-agent *task scheduling* problem, where each task τ_j of an agent A_i has to be scheduled at a specific time or time-interval. In general, a task scheduling problem consists of a set $\mathcal{A} = \{A_1, A_2, \dots A_m\}$ of m agents and a set of n tasks $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_n\}$. Each agent A_i is assumed to perform a disjoint set $\mathcal{T}_i \subseteq \mathcal{T}$ of tasks to perform. The sets $\mathcal{T}_i$ constitute a partitioning $\{\mathcal{T}_i\}_{i=1}^m$ of $\mathcal{T}$.

There exists a transitive precedence relation $\prec$ inducing a partial order on $\mathcal{T}$ signifying that $\tau_i \prec \tau_j$, whenever τ_i has to be completed before τ_j can start. We say that τ_i immediately precedes τ_j, denoted by $\tau_i \ll \tau_j$, if $\tau_i \prec \tau_j$ and there exists no τ_k such that $\tau_i \prec \tau_k$ and $\tau_k \prec \tau_j$. Furthermore, each task $\tau_i \in \mathcal{T}$ has a *processing time* p_i, a *release date* r_i (earliest allowed scheduling time), and a possibly strict *due date* d_i (latest allowed completion time).

A *solution* to this task scheduling problem is an assignment $\sigma : \mathcal{T} \to N$ assigning to each task τ_i its

starting time such that all constraints with respect to precedence relations, release dates r_i and, if they are strict, also the due dates d_i of every task are satisfied: i.e., for all $\tau_i, \tau_j \in \mathcal{T}$, we should have $\sigma(\tau_i) + p_i \leq \sigma(\tau_j)$ if $\tau_i \prec \tau_j$, and $\sigma(\tau_i) \geq r_i$ and $\sigma(\tau_i) + p_i \leq d_i$.

Sometimes we might not be satisfied with just an arbitrary assignment, but we would like to find an assignment σ minimizing the makespan, i.e., a σ such that $\max\{\sigma(\tau_i) + p_i \mid \tau_i \in \mathcal{T}\}$ is minimized.

Example 1. *Consider the running example (see Example 1).*
Here $\mathcal{T} = \{\tau_{ex_A}, \tau_{ex_B}, \tau_{ex_C}, \tau_{lun_A}, \tau_{lun_C}, \tau_{hom_A}, \tau_{hom_B}, \tau_{hom_C}, \tau_{din_B}, \tau_{cyc_C}, \tau_{idl_C}\}$*, where* τ_{ex_A} *is the first (Alice's),* τ_{ex_B} *the second (Bob's) and* τ_{ex_C} *the third (Chloe's) part of the experiment,* τ_{lun_A}*, and* τ_{lun_C} *denote the lunch of Alice and Chloe, respectively,* $\tau_{hom_A}, \tau_{hom_B}, \tau_{hom_C}$ *denote the homework of Alice, Bob and Chloe, respectively,* τ_{din_B} *is Bob's diner,* τ_{cyc_C} *Chloe's cycling and* τ_{idl_C} *denotes Chloe's forced idle time (after lunch). From the example specifications it follows that* $\tau_{ex_A} \prec \tau_{ex_B} \prec \tau_{ex_C}$*,* $\tau_{lun_A} \prec \tau_{ex_A} \prec \tau_{hom_A}$*,* $\tau_{hom_B} \prec \tau_{ex_B} \prec \tau_{din_B}$ *and* $\tau_{lun_C} \prec \tau_{idl_C} \prec \tau_{cyc_C} \prec \tau_{ex_C}$*. Furthermore, we have* $p_{ex_A} = p_{ex_B} = p_{ex_C} = 60$ *(minutes). Moreover,* $p_{lun_A} = p_{lun_C} = 30$*, while* $p_{hom_A} = p_{hom_B} = 120$*,* $p_{din_B} = 60$*,* $p_{cyc_C} = 120$ *and* $p_{idl_C} = 30$*. We also have three release dates* $r_{lun_A} = r_{hom_B} = r_{lun_C} = 0$ *and three due dates* $d_{hom_A} = d_{din_B} = d_{ex_C} = 360$*.*

Combining all the specifications results in the following conditions on a solution σ*:* $\sigma(\tau_{ex_A}) + 60 \leq \sigma(\tau_{ex_B})$*,* $\sigma(\tau_{ex_B}) + 60 \leq \sigma(\tau_{ex_C})$*,* $\sigma(\tau_{lun_A}) + 30 \leq \sigma(\tau_{ex_A})$*,* $\sigma(\tau_{ex_A}) + 60 \leq \sigma(\tau_{hom_A})$*,* $\sigma(\tau_{hom_B}) + 120 \leq \sigma(\tau_{ex_B})$*,* $\sigma(\tau_{ex_B})) + 60 \leq \sigma(\tau_{din_B})$*,* $\sigma(\tau_{lun_C}) + 30 \leq \sigma(\tau_{idl_C})$*,* $\sigma(\tau_{idl_C}) + 30 \leq \sigma(\tau_{cyc_C})$*,* $\sigma(\tau_{cyc_C}) + 120 \leq \sigma(\tau_{ex_C})$*,* $\sigma(\tau_{lun_A}) \geq 0$*,* $\sigma(\tau_{hom_B}) \geq 0$*,* $\sigma(\tau_{lun_C}) \geq 0$*,* $\sigma(\tau_{hom_A}) + 120 \leq 360$*,* $\sigma(\tau_{din_B}) + 60 \leq 360$ *and* $\sigma(\tau_{ex_C}) + 60 \leq 360$*.*

2.2 Simple Temporal Networks

To deal with task scheduling problems like the one we have presented before, Simple Temporal Networks (STNs) might be a convenient tool to use: they provide a formalism to deal with temporal variables (time-point variables) and simple constraints between them:

Definition 1. *A Simple Temporal Network* S *is a pair* (T, C)*, where* T *is a set* $\{t_0, t_1, ..., t_n\}$ *of time-point variables and* C *is a finite set of binary constraints on those variables, each constraint having the form* $t_j - t_i \leq \delta$*, for some real number* δ*. The time-point* t_0 *represents an arbitrary, fixed, reference point on the timeline. A solution* σ *to an STN* S *is an assignment of values (time-points) to time-point variables, such that all constraints are satisfied.*

As usual, we will replace the time-point t_0 by the time-point z, the zero time-point 'variable' always taking the value 0. We know that STNs can be solved efficiently: an arbitrary solution to an STN S can be found in $O(n^3)$ time [2].

Encoding a task scheduling problem as an STN is rather straightforward: Every task τ_i is represented by its starting time point t_i, whenever $\tau_i \ll \tau_j$, we have an STN constraint $p_i \leq t_j - t_i \leq \infty$ in C[2]. To encode constraints imposed by release dates, for every τ_i with release date r_i and due date d_i, we add constraints $z - t_i \leq -r_i$ and $t_i - z \leq d_i$.

Often we represent an STN $S = (T, C)$ as a directed labeled graph $G_S = (T, E, l)$ where a directed edge (t_i, t_j) is labeled by the interval $[a, b]$ if there are constraints $a \leq t_j - t_i \leq b$ in C. If we encode the running example as an STN, we get the STN shown in Figure 1. From this point on, we will denote a task by τ_i and its associated time point variable in an STN encoding by t_i.

2.3 Task scheduling problems as a special subclass of STNs

Although we can use the STN-machinery to solve task scheduling problems, we can solve an STN-encoding of a task scheduling problem in a much more efficient way. Let us denote the subclass of STNs encoding task scheduling instances by $\mathcal{STN}_{\prec}$. This subclass $\mathcal{STN}_{\prec}$ contains STNs $S = (T, C)$ characterized by the following properties:

1. for all $t_i, t_j \in T - \{z\}$, where $t_i \neq t_j$, if $a \leq t_j - t_i \leq b \in C$ then either $0 \leq a < b$ or $a < b \leq 0$; if $0 \leq a < b$ we say that t_i precedes t_j, denoted by $t_i \prec t_j$, and, analogously, if $a < b \leq 0$, we say that $t_j \prec t_i$;

[2]Sometimes, there is an additional restriction that a certain task τ_j has to be processed within b_{ij} time units once task $\tau_i \prec \tau_j$ is finished. In that case the upperbound ∞ in $p_i \leq t_j - t_i \leq \infty$ can be replaced by $b_{ij} + p_i$.

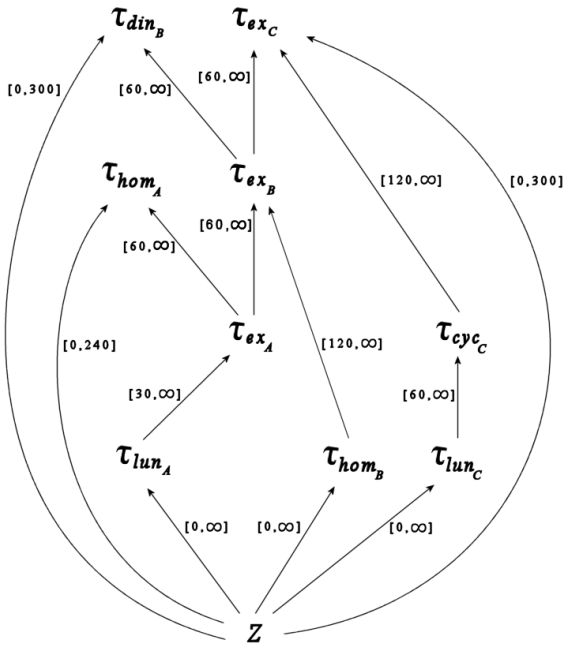


Figure 1: Alice's, Bob's and Chloe's tasks represented as an STN

2. The set $\prec$ of all such precedence constraints constitutes a partial order on T, i.e., there are no cycles in $(\mathcal{T}, \prec)$.

Let us define for every $t_i \in T$ the sets $pre(t_i) = \{t_j \in T \mid t_j \ll t_i\}$ and $suc(t_i) = \{t_j \in T \mid t_i \ll t_j\}$. Then, by the second property of $\mathcal{STN}_\prec$, every timepoint $t_i \in T$ has a disjoint set of predecessors and successors resulting in the following observation:

Observation 1. *Given an $S = (T, C) \in \mathcal{STN}_\prec$, a time-point $t_i \in T$ and a solution σ for S. If $\sigma(t_i) = s$ then the following inequalities hold:*

1. *for all $t_j \in suc(t_i)$, $\sigma(t_j) \geq s + p_i$;*
2. *for all $t_j \in pre(t_i)$, $\sigma(t_j) \leq s - p_j$.*

Given this observation and taking into account that $\sigma(t_i) \geq r_i$, we see that for every $t_i \in T$ the earliest starting time $est(t_i)$ of t_i, can be computed as follows:

$$est(t_i) = \max(\{est(t_j) + p_j \mid t_j \in pre(t_i)\} \cup \{r_i\}) \tag{1}$$

In order to compute $est(t_i)$, we only need information about the $\ll$-predecessors[3] of t_i. Hence, for every t_i we can compute $est(t_i)$ in a time that is dependent upon the size of the set $pre(t_i)$. Therefore, computing the earliest starting times $est(t_i)$ according to a $\prec$-topological ordering of T will require $O(k)$ time in total, if k is the number of edges present in the STN.

Given the deadlines d_i we can also define the latest starting time ($lst(t_i)$) of every task t_i occurring in some $S = (T, C) \in \mathcal{STN}_\prec$:

$$lst(t_i) = \min(\{lst(t_j) - p_i \mid t_j \in suc(t_i)\} \cup \{d_i - p_i\}) \tag{2}$$

Like the $est()$ computations, we see that the $lst()$ computations can also be done in $O(k)$ time.

Hence, we have the following result:

Proposition 1. *Let $S = (T, C) \in \mathcal{STN}_\prec$. Then the following holds:*

1. *for every solution σ of S and for every $t_i \in T$, $est(t_i) \leq \sigma(t_i) \leq lst(t_i)$;*
2. *an arbitrary solution σ of S can be computed incrementally in $O(k)$-time.*

This means a significant improvement above general STNs where these computations take $O(n^3)$ time.

[3] these are the immediate predecessors of t_i w.r.t. the precedence relation $\prec$.

3 Temporal Decoupling and Optimal Temporal Decoupling

The task scheduling problem we have discussed before consists of a set of agents A_i each having a set of tasks $\mathcal{T}_i$ to perform. We assume that these tasks are encoded as time-point variables in an STN $S = (T, C)$. From this point on, we will assume that each individual agent A_i has to find a solution $\sigma_i : T_i \to N$ for the STN $S_i = (T_i \cup \{z\}, C'_i)$, where T_i is the subset of tasks (time-points) assigned to A_i and C'_i is a set of constraints relevant for T_i. In doing so we have to ensure that these individual solutions σ_i together should not violate any constraint in C, that is, the merge $\sigma = \bigcup_{i=1}^{m} \sigma_i$ of all individual solutions should constitute a solution to the total problem. This problem is known as the so-called *Temporal Decoupling Problem*:

Definition 2. *Let $S = (T, C)$ be a consistent[4] STN where T is a set $\{t_0, t_1, ..., t_n\}$ of time-point variables and C is a finite set of binary constraints on those variables. Suppose that $T = \{T_i\}_{i=1}^{m}$ is partitioned in m subsets T_i. Then the temporal decoupling problem is to find m subnetworks $S_i = (T_i \cup \{z\}, C'_i)$ such that, whenever $\sigma_1, \ldots, \sigma_m$ are solutions of the individual STNs $S_1, \ldots, S_m$, respectively, their merge $\sigma = \bigcup_{i=1}^{m} \sigma_i$ is also a solution of the original STN S.*

Example 2. *In the running example, the only tasks connecting Alice and Bob are τ_{ex_A} and τ_{ex_B}. To create a subnetwork $S_A = (T_A \cup \{z\}, C'_A)$ for Alice, we take the set C_A of all constraints for the tasks of Alice and add a constraint to the latest starting time of t_{ex_A} to C_A, for example: $t_{ex_A} - z \leq 90$. To make sure that the merge of Alice's and Bob's solutions σ_A and σ_B also represents a solution σ of the original STN, we add the constraint $t_{ex_B} - z \geq 150$ to C_B. Decoupling Bob and Chloe can be done in a similar way.*

To achieve an arbitrary decoupling of an STN $S = (T, C)$ in $\mathcal{STN}_{\prec}$ where T is partitioned into blocks T_i, we have to pay attention to those tasks t_i and t_j such that $t_i \ll t_j$ and both belong to different agents A_k and A_l, respectively. Here, the problem is that agent A_k might choose a solution σ_k and A_l a solution σ_l for their own set of tasks, but thereby violate the precedence constraint $t_i \prec t_j$, i.e., it might easily occur that $\sigma_k(t_i) + p_i > \sigma_l(t_j)$.

Example 3. *Note that in our example, $\tau_{ex_A} \prec \tau_{ex_B}$. For a schedule σ_A chosen by Alice, every solution σ_A has to satisfy $\sigma(t_{ex_A}) \in [est(t_{ex_A}), lst(t_{ex_A})] = [90, 240]$ and, analogously, for Bob we have $\sigma_B(t_{ex_B}) \in [est(t_{ex_B}), lst(t_{ex_B})] = [120, 240]$. Now Alice might choose to schedule her experiment at $t_{ex_A} = 180$, while Bob independently might choose to schedule his experiment at $t_{ex_B} = 120$, which obviously conflicts with the constraint $t_{ex_A} + 60 \leq t_{ex_B}$.*

The solution ensuring that both agents can independently choose a value without violating any (inter-agent) constraint is rather simple:

1. for every such a pair of tasks t_i, t_j belonging to different agents, such that $t_i \ll t_j$, and $lst(t_i) + p_i > est(t_j)$, choose a value δ_{ij} such that $est(t_i) + p_i \leq \delta_{ij} \leq lst(t_i) + p_i$ and $est(t_j) \leq \delta_{ij} \leq lst(t_j)$.
2. we add a constraint $t_i - z \leq \delta_{ij} - p_i$ to the set of constraints in the STN $S_i = (T_i, C_i)$ of agent A_i and we add the constraint $z - t_j \leq -\delta_{ij}$ to the set of constraints of the STN $S_j = (T_j, C_j)$ of agent A_j. Here, C_i and C_j denote the restriction of the total set of constraints C to the set of constraints over $T_i \cup \{z\}$ and $T_j \cup \{z\}$, respectively.

As can be seen, this restriction will enforce any assignment σ_i chosen by A_i and σ_j chosen by A_j to satisfy $\sigma(t_i) \leq \delta_{ij} - p_i \leq \sigma(t_j) - p_i$, hence the constraint $t_i + p_i \leq t_j$ is satisfied. Note that both constraints that are added are intra-agent constraints that together imply the inter-agent constraints.

Example 4. *For a schedule σ_A chosen by Alice, for $\sigma_A(t_{ex_A})$ we have $[est(t_{ex_A}), lst(ex_A)] = [90, 240]$ and, analogously, for Bob we have $\sigma_B(t_{ex_B})$ which is restricted to $[est(t_{ex_B}), lst(t_{ex_B})] = [120, 240]$. Now Alice might choose to schedule her experiment at $t_{ex_A} = 180$, while Bob chooses to schedule his experiment at $t_{ex_B} = 120$, which conflicts with the constraint $t_{ex_A} + 60 \leq t_{ex_B}$. Decoupling would imply that we choose a value δ, for example $\delta = 200$, and we add the constraint $t_{ex_A} - z \leq 140$ to C_A and $z - t_{ex_B} \leq -200$ to C_B.*

The following observation is almost immediate

Observation 2. *An arbitrary decoupling of an STN $S = (T, C)$ in $\mathcal{STN}_{\prec}$ can be achieved in $O(k)$ time, if there are k edges present.*

[4] An STN S is called consistent if S has at least one solution.

Notice however, that the values δ_{ij} determining the decoupling are chosen arbitrarily. This is not always what we want, since agents might have task preferences that easily could be in conflict with the chosen decoupling values.

3.1 Preferences and weights for preferences

The values δ_{ij} that are chosen in decomposing an STN as discussed before are chosen arbitrarily. Some agents A_i however, might prefer some of their tasks t_i to be scheduled as early or as late as possible, while other agents might prefer to be as flexible as possible with respect to the starting time of their tasks and therefore would prefer to have available a maximal time-interval to schedule each task. In general, we assume that if agent A_i has to choose, as the result of the decoupling process, its starting time $\sigma_i(t_i)$ in an interval $[t_{i,1}, t_{i,2}]$, then the following should hold: if the agent has

1. an early preference, (s)he wants to have $t_{i,1}$ as small as possible, that is as close to $est(t_i)$ as possible;
2. a late preference, (s)he wants to have $t_{i,2}$ as large as possible, that is as close to $lst(t_i)$ as possible;
3. a flexibility preference, (s)he wants the difference $t_{i,2} - t_{i,1}$ to be as large as possible.

Example 5. *Chloe might prefer to have some time to clean and check her bike before she starts cycling, although this is not absolutely necessary for her. To get this freedom to do this, she can give a 'flexibility' preference to her cycling task. In that case, if flexibility is given (for example a time-interval of half an hour) to her in the decoupling, she can use this half hour to clean and check her bike and afterwards begin with her two hours of cycling.*

It is not difficult to see that such preferences of agents might easily conflict: Suppose we have two tasks τ_i and τ_j, with $\tau_i \ll \tau_j$. If A_i wants τ_i to be scheduled as late as possible and A_j wants τ_j to be scheduled as early as possible, we have two conflicting preferences. We could grant agent A_i his preference and ignore agent A_j or vice versa, or we could grant them each half of their preference, by choosing $t_{i,2}$ and $t_{j,1}$ such that the interval $[est(t_i), lst(t_i)]$ and the interval $[est(t_j), lst(t_j)]$ are reduced by the same amount. But what if the preference of agent A_i of task τ_i is vital to him and this is not the case for agent A_j of task τ_j? This issue can be resolved by forcing the agents to give a certain weight for their specific preference w.r.t. task τ_i indicating the importance of their preference w.r.t. τ_i. If, for example, Alice wants her experiment t_{ex_A} to start as late as possible, she can for example give a weight of 8 to her preference, while Bob might give a weight of 2 for his early preference of his experiment t_{ex_B}.

Using these weights, we can determine a *utility* for the resulting combination of a preference and it weight:

1. A task τ_i with preference 'later' and weight $l_i \geq 0$ has a utility $l_i \times (t_{i,2} - lst(t_i))$.
2. A task τ_i with preference 'early' and weight $e_i \geq 0$ contributes $e_i \times (est(t_i) - t_{i,1})$.
3. A task τ_i with preference 'flexibility' and unit value $f_i \geq 0$ contributes $f_i \times (t_{i,2} - t_{i,1})$.

Note that these utility values are dependent upon the values of the variables $t_{i,1}$ and $t_{i,2}$. Instead of constructing an arbitrary decoupling by choosing the values δ_{ij} arbitrarily, we would now like to achieve an optimal decoupling, that is, a decoupling where the sum of all the utilities of the agents is maximized[5].

Therefore we start with specifying the following linear program (LP) whose solution enables us to find the values $t_{i,1}$ and $t_{i,2}$ such that the sum of the utilities is maximized:

$$\max \sum_{i \in early} e_i \times (est(t_i) - t_{i,1}) + \sum_{i \in flex} f_i \times (t_{i,2} - t_{i,1}) + \sum_{i \in later} l_i \times (t_{i,2} - lst(t_i))$$

$$\text{s.t. } \forall i: \quad t_{i,1} \leq t_{i,2}$$
$$t_{i,2} \leq lst(t_i)$$
$$t_{i,1} \geq est(t_i)$$

$$s.t. \; \forall \tau_i \ll \tau_j,$$
$$t_j - t_i \in [a_{ij}, b_{ij}] : t_{j,2} - t_{i,1} \leq b_{ij}$$
$$t_{j,1} - t_{i,2} \geq a_{ij}$$

[5]This optimization criterion is denoted as maximizing the *utilitarian social welfare*.

Here the objective is to maximize the utilities and the inequalities guarantee that the obvious bounds for $t_{i,1}, t_{i,2}$ are respected using the lst and est values, while the last inequalities ensure that the precedence constraints are respected. The following result is easy to verify:

Proposition 2. *Let $Sol = \{t_{i,1}, t_{i,2} \mid t_i \in T\}$ be a solution of the LP stated above. Then,*

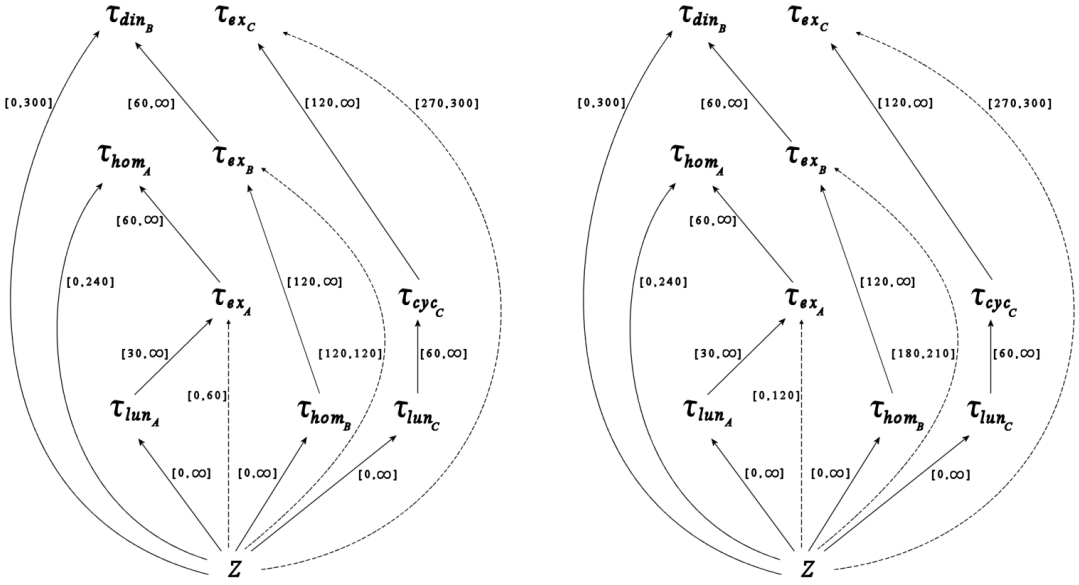
1. *if for every $t_i \ll t_j$ such that t_i belongs to A_i and t_j belongs to A_j, the intra-agent constraints $t_i \leq t_{i,2}$ and $t_j \geq t_{j,1}$ are added to the constraint sets C_i of S_i and C_j of S_j, then the STNs $S_1, S_2, \ldots, S_m$ are decoupled.*
2. *any solution σ obtained as the merge of the individual solutions σ_i optimizing the utilities of the tasks belonging to T_i is an optimal solution.*

Because we can calculate the $est(t_i)$ of all the tasks τ_i in polynomial time, and it is possible to describe the general problem with an LP formulation, we have shown that this problem is solvable in polynomial time.

Remark 1. *As has been shown in [9], optimal decoupling in STNs can be achieved in polynomial time if the objective is linear. The general construction in STNs however, requires more than $|T|^3 + |T| + |C|$ constraints and $|T|^2$ variables in the underlying LP. Using STNs in $\mathcal{S}_{\prec}$, we are able to reduce the number of constraints in the LP to $3|T| + |C|$ and the number of variables to $2|T|$.*

We remark that if all the tasks have been given an 'early' preference, we can also calculate $est(t_i)$ for every task τ_i, which gives us a decoupling in $O(k)$ time. The same holds if all the tasks have a 'later' preference, in which case we calculate $lst(t_i)$ for every task τ_i.

Example 6. *Assume that Alice wants to have lunch as early as possible (weight 3) and wants to do her experiment and homework as late as possible (weights respectively 4 and 5). Bob prefers to do his homework and experiment as early as possible (weights respectively 2 and 5), but is not certain about his plans for the evening, so he wants as much flexibility in his dinner preparation planning as possible (weight 6). Chloe wants to have lunch as early as possible (weight 2), but prefers some flexibility in her cycling schedule (weight 5) and she prefers her experiment to be as late as possible (weight 3).*
An optimal decoupling results in the subnetworks given in Figure 2(a), with total social welfare 2280.



(a) Ex.6 - Dotted lines are new constraints

(b) Ex.7 - Dotted lines are new constraints

Figure 2: Example 6 and 7 decoupled

Example 7. *Suppose every person prefers to have maximum flexibility for each task and everyone has the same weight 1. There are several optimal decouplings possible; one is given in Figure 2(b), with total social welfare equal to 390.*

Note that it is also possible to apply the LP formulation to a problem with an added optimization criterion, for example minimum makespan. In that case the task(s) τ_i with $\max\{est(t_i)+p_i \mid t_i \in T\}$ will get an extra constraint $t_{i,2} \leq est(t_i)$ and every task $\tau_j \neq \tau_i$ with $suc(t_j) = \emptyset$ will also get $t_{j,2} + p_j \leq est(t_i) + p_i$ as an extra constraint. The result is a decoupling where preferences are granted but a minimum makespan is also guaranteed.

4 Conclusion and discussion

We have shown that for task based scheduling there is a rather simple method to obtain an optimal temporal decoupling where scheduling preferences of agents have been taken into account. While in general, decoupling will take at least $O(n^3)$ time as was shown by Hunsberger, we could achieve an arbitrary decoupling in $O(k)$ time. Following a general method to obtain an optimal decoupling as has been suggested in [9], we have shown how preferences of agents can be taken into account to achieve an optimal decoupling. Although we used an LP specification of the decoupling to make the ideas underlying the decoupling clear, an algorithm has been developed based on these ideas that is able to achieve an optimal decoupling but requires less computational efforts when compared to a general LP solver if applied to this type of problem.

References

[1] Pieter Buzing, Adriaan ter Mors, Jeroen Valk, and Cees Witteveen. Coordinating self-interested planning agents. *J. of Autonomous Agents and Multi-Agent Systems*, 12:199–218, 2006.

[2] R. Dechter. *Constraint processing*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann Publishers, 2003.

[3] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[4] Marie E. desJardins, Edmund H. Durfee, Charles L. Ortiz, Jr., and Michael J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 20:13–22, 2000.

[5] Edmund H. Durfee. Distributed problem solving and planning. In Gerhard Weiss, editor, *Multiagent systems: A modern approach to distributed artificial intelligence*, pages 121–164. MIT Press, 1999.

[6] Luke Hunsberger. Algorithms for a temporal decoupling problem in multi-agent planning. In *Proceedings AAAI*, 2002.

[7] Adriaan ter Mors, Chetan Yadati, Cees Witteveen, and Yingqian Zhang. Coordination by design and the price of autonomy. *J. of Autonomous Agents and Multi-Agent Systems*, 20, 2010.

[8] Dana S. Nau. Current trends in automated planning. *AI Magazine*, 28:43–58, 2007.

[9] Léon R. Planken, Mathijs de Weerdt, and Cees Witteveen. Optimal temporal decoupling in multiagent systems. In *Proceedings AAMAS*, pages 789–796, 2010.

[10] J. Renze Steenhuisen and Cees Witteveen. Plan decoupling of agents with qualitatively constrained tasks. *Multiagent and Grid Systems*, 5:357–371, 2009.

[11] Hans Tonino, André Bos, Mathijs de Weerdt, and Cees Witteveen. Plan coordination by revision in collective agent based systems. *Artificial Intelligence*, 142:121–145, 2002.