Dynamic Temporal Decoupling

Kiriakos Simon Mountakis[1], Tomas Klos[2], and Cees Witteveen[1(✉)]

[1] Delft University of Technology, Delft, The Netherlands
C.Witteveen@tudelft.nl
[2] Utrecht University, Utrecht, The Netherlands

Abstract. Temporal decoupling is a method to distribute a temporal constraint problem over a number of actors, such that each actor can solve its own part of the problem. It then ensures that the partial solutions provided can be always merged to obtain a complete solution. This paper discusses static and dynamic decoupling methods offering maximal flexibility in solving the partial problems. Extending previous work, we present an exact $O(n^3)$ flexibility-maximizing static decoupling method. Then we discuss an exact $O(n^3)$ method for updating a given decoupling, whenever an actor communicates a commitment to a particular set of choices for some temporal variable. This updating method ensures that: (i) the flexibility of the decoupling never decreases and (ii) every commitment once made is respected in the updated decoupling. To ensure an efficient updating process, we introduce a fast heuristic to construct a new decoupling given an existing decoupling in nearly linear time. We present some experimental results showing that, in most cases, updating an existing decoupling in case new commitments for variables have been made, significantly increases the flexibility of making commitments for the remaining variables.

Keywords: Simple Temporal Problem · Decoupling · Flexibility

1 Introduction

In quite a number of cases a joint task has to be performed by several actors, each controlling only a part of the set of task constraints. For example, consider making a joint appointment with a number of people, constructing a building that involves a number of (sub)contractors, or finding a suitable multimodal transportation plan involving different transportation companies. In all these cases, some task constraints are under the control of just one actor (agent), while others require more than one agent setting the right values to the variables to satisfy them. Let us call the first type of constraints *intra-agent* constraints and the second type *inter-agent* constraints.

If there is enough time, solving such multi-actor constraint problems might involve consultation, negotiation or other agreement technologies. Sometimes, however, we have to deal with problems where communication between agents during problem solving is not possible or unwanted. For example, if the agents are in competition, there are legal or privacy issues preventing communication, or there is simply no time for communication.

D. Salvagnin and M. Lombardi (Eds.): CPAIOR 2017, LNCS 10335, pp. 328–343, 2017.
DOI: 10.1007/978-3-319-59776-8_27

In this paper we use an approach to solve multi-actor constraint problems in the latter contexts: instead of using agreement technologies, we try to avoid them by providing *decoupling techniques*. Intuitively, a decoupling technique modifies a multi-actor constraint system such that each of the agents is able to select a solution for its own set of (intra-agent) constraints and a simple merging of all individual agent solutions always satisfies the total set of constraints. Usually, this is accomplished by tightening intra-agent constraints such that inter-agent constraints are implied. Examples of real-world applications of such decoupling techniques can be found in e.g. [2,8].

Quite some research focuses on finding suitable decoupling techniques (e.g., [1,2,7]) for Simple Temporal Problems (STPs) [3]. An STP $S = (T, C)$ is a constraint formalism where a set of temporal variables $T = \{t_0, t_1, \ldots, t_n\}$ are subject to binary difference constraints C and solutions can be found in low polynomial ($O(n^3)$) time. Since STPs deal with temporal variables, a decoupling technique applied to STPs is called *temporal decoupling*. The quality of a temporal decoupling technique depends on the degree to which it tightens intra-agent constraints: the more it restricts the flexibility of each individual agent in solving their own part of the constraints, the less it is preferred. Therefore, we need a flexibility metric to evaluate the quality of a temporal decoupling.

Originally, flexibility was measured by summing the differences between the highest possible (latest) and the lowest possible (earliest) values of all variables in the constraint system after decoupling ([6,7]). This so-called *naive flexibility metric* has been criticized, however, because it does not take into account the dependencies between the variables and, in general, seriously overestimates the "real" amount of flexibility. An alternative metric, the *concurrent flexibility metric* has been proposed in [15]. This metric accounts for dependencies between the variables and is based on the notion of an *interval schedule* for an STP S: For each variable $t_i \in T$ an interval schedule defines an interval $[l_i, u_i]$, such that choosing a value within $[l_i, u_i]$ for each variable t_i, always constitutes a solution for S. The sum $\sum_{i=1}^{n}(u_i - l_i)$ of the widths of these intervals determines the flexibility of S. The concurrent flexibility of S then is defined as the maximum flexibility we can obtain for an interval schedule of S and can be computed in $O(n^3)$ (see [9]).

As shown in [15], the concurrent flexibility metric can also be used to obtain an optimal (i.e. maximum flexible) temporal decoupling of an STP. This decoupling is a *total decoupling*, that is, a decoupling where the n variables are evenly distributed over n independent agents and thus every agent controls exactly one variable. It has been shown that this total decoupling is optimal for every partitioning of the set of temporal variables if one considers the concurrent flexibility metric as the flexibility metric to be used. In this paper, we concentrate on such (optimal) total decouplings of an STP.

In all existing approaches, a single temporal decoupling is computed in advance and is not changed, even if later on some agents announce their commitment to a specific value (or range of values) for a variable they control. Intuitively, however, we can use such additional information for the benefit of all

agents, by possibly increasing the flexibility of variables they are not yet committed to. Specifically, when an agent announces a commitment to a sub-range of values within the given interval schedule (that represents the current decoupling), we are interested in updating the decoupling such that the individual flexibility of no agent is affected negatively.

More precisely, the overall aim of this paper is to show that a decoupling update method with the following nice properties do exist: first of all, it never affects the current flexibility of the agents negatively, and, secondly, it never decreases (and possibly increases) the individual flexibility of the variables not yet committed to. We will also show that updating a temporal decoupling as the result of a commitment for a single variable can be done in almost linear (amortized) time ($O(n \log n)$), which compares favourably with the cost of computing a new optimal temporal decoupling ($O(n^3)$).

Organisation. In Sect. 2 we discuss existing relevant work on STPs and temporal decoupling (TD). Then, in Sect. 3, extending some existing work, we briefly show how a total TD can be computed in $O(n^3)$, using a minimum matching approach. In Sect. 4, we first provide an exact approach to update an existing decoupling after commitments to variables have been made. We conclude, however, that this adaptation is computationally quite costly. Therefore, in Sect. 5 we offer an alternative, heuristic method, that is capable to adapt a given temporal decoupling in almost linear time per variable commitment. To show the benefits of adapting the temporal decoupling, in Sect. 6 we present the results of some experiments using STP benchmarks sets with the heuristic decoupling update and compare the results with an exact, but computationally more intensive updating method. We end with stating some conclusions and suggestions for further work.

2 Preliminaries

Simple Temporal Problems. A Simple Temporal Problem (STP) (also known as a Simple Temporal Network (STN)) is a pair $S = (T, C)$, where $T = \{t_0, t_1, \ldots, t_n\}$ is a set of temporal variables (events) and C is a finite set of binary difference constraints $t_j - t_i \leq c_{ij}$, for some real number c_{ij}.[1] The problem is to find a solution, that is a sequence $(s_0, s_1, s_2, \ldots, s_n)$ of values such that, if each $t_i \in T$ takes the value s_i, all constraints in C are satisfied. If such a solution exists, we say that the STN is *consistent*. [2] In order to express absolute time constraints, the time point variable $t_0 \in T$, also denoted by z, is used. It represents a fixed reference point on the timeline, and is always assigned the value 0.

[1] If both $t_j - t_i \leq c_{ij}$ and $t_i - t_j \leq c_{ji}$ are specified, we sometimes use the more compact notations $-c_{ji} \leq t_j - t_i \leq c_{ij}$ or $t_j - t_i \in [-c_{ji}, c_{ij}]$.

[2] W.l.g., in the remainder of the paper we simply assume an STN to be consistent. Consistency of an STN can be determined in low-order polynomial time [4].

Example 1. Consider two trains 1 and 2 arriving at a station. Train 1 has to arrive between 12:05 and 12:15, train 2 between 12:08 and 12:20. People traveling with these trains need to change trains at the station. Typically, one needs at least 3 min for changing trains. Train 1 will stay during 5 min at the station and train 2 stays 7 min before it departs. Let t_i denote the arrival time for train $i = 1, 2$. Let $t_0 = z = 0$ represent noon (12:00). To ensure that all passengers have an opportunity to change trains, we state the following STP $S = (T, C)$ where: $T = \{z, t_1, t_2\}$, and $C = \{5 \leq t_1 - z \leq 15,\ 8 \leq t_2 - z \leq 20,\ -2 \leq t_2 - t_1 \leq 4\}$. As the reader may verify, two possible solutions[3] for this STP are $s = (0, 10, 10)$ and $s' = (0, 15, 17)$. That is, there is a solution when both trains arrive at 12:10, and there is also a solution where train 1 arrives at 12:15, while train 2 arrives at 12:17. ■

An STP $S = (T, C)$ can also be specified as a directed weighted graph $G_S = (T_S, E_S, l_S)$ where the vertices T_S represent variables in T and for every constraint $t_j - t_i \leq c_{ij}$ in C, there is a directed edge $(t_i, t_j) \in E_S$ labeled by its weight $l_S(t_i, t_j) = c_{ij}$. The weight c_{ij} on the arc (t_i, t_j) can be interpreted as the length of the path from t_i to t_j. Using a shortest path interpretation of the STP $S = (T, C)$, there is an efficient method to find all shortest paths between pairs (t_i, t_j) using e.g. Floyd and Warshall's all-pairs shortest paths algorithm [5]. A shortest path between time variables t_i and t_j then corresponds to a *tightest constraint* between t_i and t_j. These tightest constraints can be collected in an $n \times n$ *minimum distance matrix* $D = [d_{ij}]$, containing for every pair of time-point variables t_i and t_j the length d_{ij} of the *shortest path* from t_i to t_j in the distance graph. In particular, the first row and the first column of the distance matrix D contain useful information about the possible schedules for S: The sequence $lst = (d_{00}, d_{01}, \ldots, d_{0n})$ specifies the latest starting time solution for each time point variable t_i. Analogously, $est = (-d_{00}, -d_{10}, \ldots, -d_{no})$ specifies the *earliest starting time* solution.

Example 2. Continuing Example 1, the minimum distance matrix D of S equals

$$D = \begin{bmatrix} 0 & 15 & 19 \\ -5 & 0 & 4 \\ -8 & 2 & 0 \end{bmatrix}$$

The earliest time solution therefore is $est = (0, 5, 8)$, and the latest time solution $lst = (0, 15, 19)$. ■

Given S, the matrix D can be computed in low-order polynomial ($O(n^3)$) time, where $n = |T|$, see [3].[4] Hence, using the STP-machinery we can find earliest and latest time solutions quite efficiently.

[3] Of course, there are many more.

[4] An improvement by Planken et al. [12] has shown that a schedule can be found in $O(n^2 w_d)$-time where w_d is the graph width induced by a vertex ordering.

Temporal Decoupling. In order to find a solution for an STP $S = (T, C)$ all variables $t_i \in T$ should be assigned a suitable value. Sometimes these variables are controlled by different agents. That is, $T - \{z\} = \{t_1, \ldots, t_n\}$ is partitioned into k non-empty and non-overlapping subsets $T_1, \ldots, T_k$ of T, each T_j corresponding to the set of variables controlled by agent $a_j \in A = \{a_1, a_2, \ldots, a_k\}$. Such a partitioning of $T - \{z\}$ will be denoted by $[T_i]_{i=1}^k$. In such a case, the set of constraints C is split in a set $C_{intra} = \bigcup_{i=1}^k C_i$ of intra-agent constraints and a set $C_{inter} = C - C_{intra}$ of inter-agent constraints. Here, a constraint $t_i - t_j \leq c_{ji}$ is an intra-constraint occurring in C_h if there exists a subset T_h such that $t_i, t_j \in T_h$, else, it is an inter-agent constraint. Given the partitioning $[T_j]_{j=1}^k$, every agent a_i completely controls the (sub) STP $S_i = (T_i \cup \{z\}, C_i)$, where C_i is its set of intra-agent constraints, and determines a solution s_i for it, independently from the others. In general, however, it is not the case that, using these sub STPs, merging partial solutions s_i will always constitute a total solution to S:

Example 3. Continuing Example 1, let train i be controlled by agent a_i for $i = 1, 2$ and assume that we have computed the set of tightest constraints based on C. Then $S_1 = (\{z, t_1\}, \{5 \leq t_1 - z \leq 15\})$ and $S_2 = (\{z, t_2\}, \{8 \leq t_2 - z \leq 19\})$. Agent a_1 is free to choose a time t_1 between 5 and 15 and suppose she chooses 10. Agent a_2 controls t_2 and, therefore, can select a value between 8 and 19. Suppose he chooses 16. Now clearly, both intra-agent constraints are satisfied, but the inter-agent constraint $t_2 - t_1 \leq 4$ is not, since $16 - 10 > 4$. Hence, the partial solutions provided by the agents are not conflict-free. ■

The *temporal decoupling* problem for $S = (T, C)$, given the partitioning $[T_j]_{j=1}^k$, is to find a suitable set $C'_{intra} = \bigcup_{i=1}^k C'_i$ of (modified) intra-agent constraints such that if s'_i is an arbitrary solution to $S'_i = (T_i \cup \{z\}, C'_i)$, it always can be merged with other partial solutions to a total solution s' for S.[5]

We are interested, however, not in arbitrary decouplings, but an *optimal decoupling* of the k agents, allowing each agent to choose an assignment for its variables independently of others, while maintaining *maximum flexibility*. This optimal decoupling problem has been solved in [14] for the total decoupling case, that is the case where $k = n$ and each agent a_i controls a single time point variable t_i. In this case, the decoupling results in a specification of a lower bound l_i and an upper bound u_i for every time point variable $t_i \in T$, such that t_i can take any value $v_i \in [l_i, u_i]$ without violating any of the intra- or inter-agent constraints. This means that if agent a_i controls T_i then her set of intra-agent constraints is $C_i = \{l_j \leq t_j \leq u_j \mid t_j \in T_i\}$.

The total flexibility the agents have, due to this decoupling, is determined by the sum of the differences $(u_i - l_i)$. Therefore, the decoupling bounds $l = (l_i)_{i=1}^n$ and $u = (u_i)_{i=1}^n$ are chosen in such a way that the flexibility $\sum_i (u_i - l_i)$ is maximized. It can be shown (see [14]) that such a pair (l, u) can be obtained as a maximizer of the following LP:

[5] In other words, the set $\bigcup_{i=1}^k C'_i$ logically implies the set C of original constraints.

$$\max_{l,u} \sum_i (u_i - l_i) \qquad (\mathbf{TD}(D))$$

$$\begin{aligned} \text{s.t.} \quad & l_0 = u_0 = 0 && (1)\\ & l_i \le u_i \qquad \forall i \in T && (2)\\ & u_j - l_i \le d_{ij} \qquad \forall i \ne j \in T && (3) \end{aligned}$$

where D is the minimum distance matrix associated with S.

Example 4. Consider the matrix D in Example 2 and the scenario sketched in Example 3. Then the LP whose maximizers determine a maximum decoupling is the following:

$$\begin{aligned} \max_{l,u} \quad & \sum_i (u_i - l_i)\\ \text{s.t.} \quad & u_0 = l_0 = 0,\\ & l_1 \le u_1, \quad l_2 \le u_2\\ & u_1 - l_0 \le 15, \quad u_2 - l_0 \le 19\\ & u_0 - l_1 \le -5, \quad u_2 - l_1 \le 4\\ & u_0 - l_2 \le -8, \quad u_1 - l_2 \le 2 \end{aligned}$$

Solving this LP, we obtain $\sum_{i=0}^{2}(u_i - l_i) = 6$ with maximizers $l = (0, 15, 13)$ and $u = (0, 15, 19)$. This implies that in this decoupling (l, u) agent a_1 is forced to arrive at 12:15, while agent a_2 can choose to arrive between 12:13 and 12:19. ■

Remark. Note that the total decoupling solution (l, u) for S also is a solution for a decoupling based on an arbitrary partitioning $[T_i]_{i=1}^{k}$ of S. Observe that (l, u) is a decoupling, if for any value v_i chosen by a_h and any value w_j chosen by $a_{h'}$ for every $1 \le h \ne h' \le k$, we have $v_i - w_j \le d_{ji}$ and $w_j - v_i \le d_{ij}$. Since (l, u) is a total decoupling, it satisfies the conditions of the LP $\mathbf{TD}(D)$. Hence, $u_i - l_j \le d_{ij}$ and $u_j - l_i \le d_{ji}$. Since $v_i \in [l_i, u_i]$ and $w_j \in [l_j, u_j]$, we then immediately have $v_i - w_j \le u_i - l_j \le d_{ji}$ and $w_j - v_i \le u_j - l_i \le d_{ji}$. Therefore, whatever choices are made by the individual agents satisfying their local constraints, these choices always will satisfy the original constraints, too. ■

Remark. In [14] the (l, u) bounds found by solving the LP $\mathbf{TD}(D)$ are used to compute the concurrent flexibility $\mathit{flex}(S) = \sum_i^n (u_i - l_i)$ of an STP S. Taking the concurrent flexibility as our flexibility metric, the (l, u) bounds for decoupling are always optimal, whatever partitioning $[T_i]_{i=1}^{n}$ is used: first, observe that due to a decoupling, the flexibility of an STN can never increase. Secondly, if the (l, u) bounds for a total decoupling are used, by definition, the sum $\sum_{i=1}^{k} \mathit{flex}(S_i)$ of the (concurrent) flexibilities of the decoupled subsystems equals the flexibility $\mathit{flex}(S)$ of the original system. Hence, the total decoupling bounds (l, u) are optimal, whatever partitioning $[T_i]_{i=1}^{n}$ used. ■

In this paper, we will consider concurrent flexibility as our flexibility metric. Hence, a total decoupling is always an optimal decoupling for any partitioning of variables. Therefore, in the sequel, we concentrate on total decouplings.

3 Total Decoupling by Minimum Matching

In the introduction we mentioned that an (optimal) total decoupling can be achieved in $O(n^3)$ time. In the previous section, we presented an LP to compute such a decoupling. If the STP has n variables, the LP to be solved has $2n$ variables and n^2 constraints. Modern interior-point methods solve LPs with n variables and m constraints in roughly $O(n^3 m)$ [13]. Thus, the complexity of solving total decoupling as an LP might be as high as $O(n^5)$. In a previous paper ([9]), we have shown that computing the concurrent flexibility of an STP can be reduced to a *minimum matching* problem (see [10]) using the distance matrix D to construct a weighted cost matrix D^* for the latter problem.

This reduction, however, does not allow us to directly compute the corresponding flexibility maximizers (l, u). In this section we therefore show that there is a full $O(n^3)$ alternative method for the LP-based flexibility method to compute the concurrent flexibility and the corresponding maximizers (l, u), thereby showing that an optimal total decoupling can be computed in $O(n^3)$.

Flexibility and Minimum Matching. Given an STN $S = (T, C)$ with minimum distance matrix D, let $\mathit{flex}(S)$ be its concurrent flexibility, realised by the maximizers (l, u). Hence, $\mathit{flex}(S) = f(l, u) = \sum_{i=1}^{n}(u_i - l_i)$. Unfolding this sum –as was noticed in [9]– we obtain

$$f(l,u) = \sum_{i \in T}(u_i - l_i) = \sum_{i \in T}(u_{\pi_i} - l_i) \tag{4}$$

for every permutation π of T.[6] Since (l, u) is a total decoupling, we have

$$u_j - l_i \leq d_{ij} \qquad \forall i \neq j \in T \tag{5}$$

$$u_j - l_j = (u_j - z) + (z - l_j) \leq d_{0j} + d_{j0} \qquad \forall j \in T \tag{6}$$

Hence, defining the modified distance matrix (also called *weight matrix*) $D^* = [d^*_{ij}]_{n \times n}$ by

$$d^*_{ij} = \begin{cases} d_{ij}, & 1 \leq i \neq j \leq n \\ d_{0i} + d_{i0}, & 1 \leq i = j \leq n \end{cases}$$

we obtain the following inequality:

$$f(l,u) \leq \min\{\sum_{i \in T} d^*_{i\pi_i} : \pi \in \Pi(T)\} \tag{7}$$

[6] To avoid cumbersome notation, we will often use $i \in T$ as a shorthand for $t_i \in T$.

where $\Pi(T)$ is the space of permutations of T. Equation (7) states that the maximum flexibility of an STN is upper bounded by the value of a minimum matching in a bipartite graph with weight matrix D^*. The solution of such a matching problem consists in finding for each row i in D^* a unique column π_i such that the sum $\sum_{i \in T} d^*_{i\pi_i}$ is minimized. As we showed in [9], by applying LP-duality theory, Eq. (7) can be replaced by an equality: $f(l,u) = \min_{\pi \in \Pi(T)} \sum_{i \in T} d^*_{i\pi_i}$, so the concurrent flexibility $\mathit{flex}(S) = f$ can be computed by a minimum matching algorithm as e.g. the Hungarian algorithm, running in $O(n^3)$ time.

Finding a Maximizer (l, u) Using Minimum Matching. With the further help of LP-duality theory i.c., complementary slackness conditions [10], the following result is immediate:

Observation 1. *If π is a minimum matching with value m for the weight matrix D^*, then there exists a maximizer (l,u) for the concurrent flexibility $\mathit{flex}(S)$ of S, such that $\mathit{flex}(S) = m$ and for all $i \in T$, $u_{\pi_i} - l_i = d^*_{i\pi_i}$.*

Now observe that the inequalities stated in the LP-specification $\mathbf{TD}(D)$ and the inequalities $u_{\pi_i} - l_i = d^*_{i\pi_i}$ in Observation 1 all are binary difference constraints. Hence, the STP $S' = (T', C')$, where

$$\begin{aligned} T' &= L \cup U = \{l_i \mid i \in T\} \cup \{u_i \mid i \in T\}, \\ C' &= \{u_i - l_j \leq d^*_{ij} \mid i,j \in T\} \cup \{l_i - u_{\pi_i} \leq -d^*_{i\pi_i} \mid i \in T\} \cup \{l_i - u_i \leq 0 \mid i \in T\} \end{aligned}$$

is an STP[7] and every solution $s' = (l_1, \ldots l_n, u_1, \ldots, u_n)$ of S' in fact is a maximizer (l,u) for the original STP S, since the flexibility associated with such a solution (l,u) satisfies $\mathit{flex}(S) \geq f(l,u) = \sum_{i \in T}(u_i - l_i) \geq \sum_{i \in T} d^*_{i\pi_i} = \mathit{flex}(S)$. Hence, this pair (l,u) is a maximizer realizing $\mathit{flex}(S)$.

In particular, the earliest and latest solutions of S' have this property. Hence, since (i) D^* can be obtained in $O(n^3)$ time, (ii) a minimum matching based on D^* can be computed in $O(n^3)$, and (iii) the STN S' together with a solution $s = (l,u)$ for it can be computed in $O(n^3)$, we obtain the following result:

Corollary 1. *Given an STN $S = (T, C)$ with distance matrix D, an optimal total decoupling (l,u) for S can be found in $O(n^3)$.*

4 Dynamic Decoupling by Updating

A temporal decoupling allows agents to make independent choices or commitments to variables they control. As pointed out in the introduction, we want to adapt an existing (total) temporal decoupling (l,u) whenever an agent makes a new commitment to one or more temporal variables (s)he controls. To show that such a commitment could affect the flexibility other agents have in making their possible commitments, consider our leading example again:

[7] We should note that this STP has two external reference variables $u_0 = l_0 = 0$.

Example 5. In Example 3 we obtained a temporal decoupling $(l, u) = ((0, 15, 13), (0, 15, 19))$. Here, agent 1 was forced to arrive at 12:15, but agent 2 could choose to arrive between 12:13 and 12:19. Suppose agent 2 commits to arrive at 12:13. As a result, agent 1 is able to arrive at any time in the interval $[9, 15]$. Then, by adapting the decoupling to the updated decoupling $(l', u') = ((0, 9, 13), (0, 15, 13))$, the flexibility of agent 1 could increase from 0 to 6, taking into account the new commitment agent 1 has made. If the existing commitment (l, u), however, is not updated, agent 1 still has to choose 12:15 as its time of arrival. ■

In order to state this *dynamic decoupling* or *decouple updating* problem more precisely, we assume that at any moment in time the set T consists of a subset T_c of variables committed to and a set of not committed to, or *free*, variables T_f. The commitments for variables $t_i \in T_c$ are given as bounds $[l_i^c, u_i^c]$, specifying that for $i \in T_c$, t_i is committed[8] to choose a value in the interval $[l_i^c, u_i^c]$. The total set of commitments in T_c is given by the bounds (l^c, u^c). We assume that these committed bounds do not violate decoupling conditions.

Whenever (l, u) is a total decoupling for $S = (T, C)$, where $T = T_c \cup T_f$, we always assume that (l, u) respects the commitments, i.e. $[l_i, u_i] = [l_i^c, u_i^c]$ for every $t_i \in T_c$, and (l, u) is an optimum decoupling for the free variables in T_f, given these commitments. Suppose now an agent makes a new commitment for a variable $t_i \in T_f$. In that case, such a commitment $[v_i, w_i]$ should satisfy the existing decoupling, that is $l_i \leq v_i \leq w_i \leq u_i$, but as a result, the new decoupling where $[l_i, u_i] = [v_i, w_i]$, $T_c' = T_c \cup \{t_i\}$, and $T_f' = T_f - \{t_i\}$ might no longer be an optimal decoupling w.r.t. T_f' (e.g. see Example 5). In that case we need to update (l, u) and to find a better decoupling (l', u').

The decoupling updating problem then can be stated as follows:

> Given a (possibly non-optimal) total decoupling (l, u) for an STP $S = (T, C)$, with $T = T_c \cup T_f$, find a new total decoupling (l', u') for S such that
>
> 1. no individual flexibility of free variables is negatively affected, i.e., for all $j \in T_f$, $[l_j, u_j] \sqsubseteq [l_j', u_j']$[9];
> 2. all commitments are respected, that is, for all $j \in T_c$, $[l_j'^c, u_j'^c] = [l_j^c, u_j^c]$;
> 3. the flexibility realized by (l', u'), given the commitments (l'^c, u'^c), is maximum.

Based on the earlier shown total decoupling problem $\mathbf{TD}(D)$, this decoupling update problem can also be stated as the following LP:

$$\max \quad \sum_j (u_j' - l_j') \qquad (\mathbf{DTD}(D, T_c, T_f, (l, u))$$

[8] Note that this is slightly more general concept than a strict commitment of a variable t_i to one value v_i.

[9] That is, the interval $[l_j', u_j']$ contains $[l_j, u_j]$.

$$\text{s.t.} \quad u'_0 = l'_0 = 0 \tag{8}$$
$$u'_j - l'_i \leq d_{ij} \qquad \forall i \neq j \in T \tag{9}$$
$$u_j \leq u'_j,\ l'_j \leq l_j \qquad \forall j \in T_f \tag{10}$$
$$l'_j = l_j,\ u'_j = u_j \qquad \forall j \in T_c \tag{11}$$

Here, (l, u) is the existing total decoupling.

In fact, by transforming **DTD**-instances into **TD**-instances, we can show that the dynamic decoupling (decoupling updating) problem can be reduced to a minimum-matching problem and can be solved in $O(n^3)$, too.[10]

5 A Fast Heuristic for Updating

Although the dynamic total decoupling problem can be reduced to the static temporal decoupling problem, in practice, the computational complexity involved might be too high. While an initial decoupling might be computed off-line, an update of a decoupling requires an on-line adaptation process. Since the costs of solving such a dynamic matching problem are at least $O(n^2)$ per update, we would like to alleviate this computational investment. Therefore, in this section we discuss a fast heuristic for the decoupling updating problem. Using this heuristic, we show that an updated decoupling can be found in (amortized) $O(n \log n)$ per update step if $O(n)$ updates are assumed to take place.

The following proposition is a simple result we base our heuristic on:

Proposition 1. *If (l, u) is a total decoupling for S with associated weight matrix D^*, then, for all $j \in T$, $l_j = \max_{k \in T}(u_k - d^*_{kj})$ and $u_j = \min_{k \in T}(l_k + d^*_{jk})$.*

Proof. Since (l, u) is a maximizer of the LP $\mathbf{TD}(D)$, $u_i - l_j \leq d^*_{ij}$, for every $i, j \in T$. Hence, for each $i, j \in T$, we have $l_j \geq \max_{k \in T}(u_k - d^*_{kj})$ and $u_i \leq \min_{k \in T}(l_k + d^*_{ik})$. Now, on the contrary, assume that for some $i \in T$, $l_i > \min_{k \in T}(u_k - d^*_{ki})$. Then the bounds (l', u) where $l' = l$, except for $l'_i = \min_{k \in T}(u_k - d^*_{ki})$, would satisfy the constraints $u_i - l'_j \leq d^*_{ij}$ for every $i, j \in T$, as well as $l'_j \leq u_j$ for every $j \in T$. Hence, (l', u) is a decoupling as well. But then (l', u) satisfies the conditions of the LP $\mathbf{TD}(D)$ but $\sum_j (u_j - l'_j) > \sum_j (u_j - l_j)$, contradicting the fact that (l, u) is a maximizer of this LP. Hence, such an $i \in T$ cannot exist. The proof for $u_i < \min_{k \in T}(l_k + d_{ik})$ goes along the same line and the proposition follows. ∎

The converse, however, of this proposition is not true: not every solution (l, u) satisfying the two equalities is a maximum decoupling. It can only be shown that in such a case (l, u) is a *maximal* total decoupling. That means, if (l, u) satisfies

[10] As has been observed by a reviewer, there exists an $O(n^2)$ algorithm for the dynamic variant of the minimum-matching problem. It is likely that our dynamic decoupling problem can be solved by such a dynamic minimum matching algorithm in $O(n^2)$ time, too.

the equations above, there does not exist a decoupling (l', u') containing (l, u) that has a higher flexibility.

It turns out that these *maximal total decouplings* and their updates can be computed very efficiently: given a (non-maximal) total decoupling (l, u), and a set $T = T_c \cup T_f$ of committed and free variables, there exists a very efficient algorithm to compute a new total decoupling $[l', u']$ such that

1. (l', u') preserves the existing commitments: for all $j \in T_c$, $[l'_j, u'_j] = [l_j, u_j]$;
2. $[l', u']$ respects the existing bounds of the free variables: for all $j \in T_f$, $[l_j, u_j] \sqsubseteq [l'_j, u'_j]$;
3. (l', u') satisfies the conditions stated in Proposition 1 above w.r.t. the free variables, i.e. (l', u') is a maximal decoupling with respect to variables in T_f.

The following surprisingly simple algorithm finds a maximal flexible total decoupling for a set $T_f \cup T_c$ of free and committed temporal variables that satisfies these conditions. The algorithm iteratively updates the existing (l, u)-decoupling bounds for the variables in T_f until all free variables satisfy the equations stated in Proposition 1.

Algorithm 1. Finding an update (l', u') of an existing total decoupling (l, u)

Require: (l, u) is a total decoupling for S; $D^* = [d^*_{ij}]$ is the weight matrix; $T = T_f \cup T_c$;
1: $l' = l$; $u' = u$;
2: **for** $i = 1$ to $|T_f|$ **do**
3: $\quad \min_i := \max_{k \in T}(u'_k - d^*_{ik})$;
4: $\quad \max_i := \min_{k \in T}(l'_k + d^*_{kj})$;
5: $\quad$ **if** $l'_i > \min_i$ **then**
6: $\quad\quad l'_i \leftarrow \min_i$
7: $\quad$ **if** $u'_i < \max_i$ **then**
8: $\quad\quad u'_i \leftarrow \max_i$
9: **return** (l', u');

It is easy to see that the algorithm preserves the existing commitments for variables in T_c, since only bounds of free variables in T_f are updated.

It is also easy to see that the existing bounds $[l_j, u_j]$ of free variables $j \in T_f$ are respected: For every j it holds that either $u_j < \max_j$ ($l'_j > \min_j$, respectively) or $u_j = \max_j$ ($l'_j > \min_j$, respectively). Hence, $u'_j \geq u_j$ and $l'_j \leq l_j$.

To show that the obtained decoupling (l', u') is maximal with respect to the free variables, we state two key observations: first of all, in every step i it holds that $l'_i \geq \min_i$ and $u'_i \leq \max_i$, because (l, u) is a decoupling and the $(\min_i, \max_i)$ bounds are not violated during updating. Secondly, once the bounds (l'_i, u'_i) for a variable $i \in T_f$ have been updated to $\min_i$ and $\max_i$, (l'_i, u'_i) will never need to be updated again, since $\min_i$ depends on values u'_k that might increase or stay the same; hence $\min_i$ cannot decrease in subsequent steps. Likewise, $\max_i$ depends on values l'_k that might decrease or stay the same. Therefore, $\max_i$ cannot increase in later steps. Therefore, it is sufficient to update the bounds for the variables only once.

As a result, at the end all free variables have been updated and achieved their minimal/maximal bound. Then a maximal total decoupling has been obtained, since all variables in T_f will satisfy the equations stated in Proposition 1.

Example 6. Continuing our example, notice that the weight matrix D^* obtained via the minimum distance matrix D (see Example 2) equals

$$D^* = \begin{bmatrix} 0 & 15 & 19 \\ -5 & 10 & 4 \\ -8 & 2 & 6 \end{bmatrix}$$

Given the decoupling $(l, u) = ((0, 15, 13), (0, 15, 19))$ with $T_f = \{t_1, t_2\}$, let agent 2 commit to $t_2 = 13$. We would like to compute a new decoupling maximizing the flexibility for t_1. Note that $\min_1 = \max\{5, 5, 9\} = 9$ and $\max_1 = \min\{15, 25, 15\} = 15$. Hence, the heuristic finds an updated decoupling $(l', u') = ((0, 9, 13), (0, 15, 13))$.

Complexity of the Heuristic. As the reader quickly observes, a naive implementation of Algorithm 1 would require $O(n^2)$-time to obtain an updated decoupling: To update a single variable $i \in T_f$, we have to compute $\min_i$ and $\max_i$. This costs $O(n)$ per variable and there may be n variables to update.

Fortunately, if there are $O(n)$ updating steps, the computational cost per step can be significantly reduced: First compute, for each $i \in T$, a priority queue $Q_{\min(i)}$ of the entries $u_k - d^*_{ik}$, $k \in T$, and a priority queue $Q_{\max(i)}$ of the entries $l_k + d^*_{ki}$, $k \in T$. The initialisation of these priority queues will cost $O(n \,.\, n \log n) = O(n^2 . \log n)$.

After a new commitment for a variable j, say $[v_j, w_j]$, we have to compute a decoupling update. It suffices to update the priority queues $Q_{\min(i)}$ and $Q_{\max(i)}$ for every $i \in T_f$. In this case, the element $u_j - d^*_{ij}$ in the queue $Q_{min(i)}$ has to be changed to $w_j - d^*_{ij}$ and the element l_j in queue $Q_{max(i)}$ has to be changed to $v_j + d^*_{ji}$. Maintaining the priority order in the priority queues will cost $O(\log n)$ per variable. Hence, the total cost for computing a new decoupling are $O(n \log n)$. If there are $O(n)$ updates, the total cost of performing these $O(n)$ updates are $O(n^2 . \log n) + O(n).O(n . \log n) = O(n^2 . \log n)$. Hence, the amortized time costs per update are $O(n . \log n)$.

In the next section we will verify the quality of such maximal flexible decouplings experimentally, comparing them with maximum flexible total decouplings and a static decoupling.

6 Experimental Evaluation

We discuss an experimental evaluation of the dynamic total decoupling method. These experiments have been designed (i) to verify whether updating a decoupling indeed significantly increases the decoupling flexibility compared to using a static decoupling and (ii) to verify whether the heuristic significantly reduces the computational investments in updating as expected.

Material. We applied our updating method to a dataset of STP benchmark instances (see [11]). This dataset contains a series of sets of STP-instances with STPs of varying structure and number of variables. Table 1 contains the main characteristics of this benchmark set. The size of the STP-instances in this dataset varies from instances with 41 variables with 614 constraints to instances with 4082 nodes and 14110 constraints. In total, these sets contain 1122 STP-instances. We used MATLAB (R2016b) on an iMac 3.2 Ghz, Intel Core i5, with 24 Gb memory to perform the experiments.

Table 1. STP Benchmarksets used in the experiments.

Benchmark set	NR instances	Min vars	AV vars	Max vars
bdh	300	41	207	401
Diamonds	130	111	857	2751
Chordalgraphs	400	200	200	200
NeilYorkeExamples	180	108	1333	4082
sf-fixedNodes	112	150	150	150

Method. For each instance in the benchmark set we first computed a maximum decoupling (l, u) with its flexibility $flex = \sum_{j=1}^{n}(u_j - l_j)$, using the minimum matching method. Then, according to an arbitrary ordering $< t_1, t_2, \ldots t_n >$ of the variables in T, each variable t_i is iteratively committed to a fixed value v_i, where v_i is randomly chosen in an interval (l_i, u_i) belonging to the current temporal decoupling (l, u). After each such a commitment, we compute a new updated decoupling (l^i, u^i) where $T_c = \{t_1, \ldots t_i\}$ and $T_f = \{t_{i+1}, \ldots t_n\}$. The total flexibility of the new decoupling (l^i, u^i) is now dependent upon the $n-i$ free variables in T_f and will be denoted by f_h^i. We initially set $f_h^0 = flex$. In order to account for the decreasing number of free variables after successive updates, we compute after each update the heuristic update flexibility per free variable in T_f: $f_h^i/(n-i)$. To compare these flexibility measures with the static case, for the latter we take the total flexibility of the free variables $flex^i = \sum_{j=i+1}^{n}(u_j - l_j)$ and then compute the flexibility per free time variable without updating: $flex^i/(n-i)$.

As a summary result for each benchmark instance k, we compute

1. the average over all updates of the static flexibility per free variable: $av\_stat = \sum_{i=1}^{n} flex^i/((n-i) \cdot n)$
2. the average over all updates of the heuristic update flexibility per free variable: $av\_h = \sum_{i=1}^{n} f_h^i/((n-i) \cdot n)$

Note that the ratio $rel\_flex_h = av\_h/av\_stat$ indicates the impact of the heuristic updating method for a particular instance: a value close to 1 indicates almost no added flexibility (per time variable) by updating, but a value of 2 indicates that the flexibility (per time variable) doubled by updating the decoupling.

Finally, we collected the rel_flex_h results per instance for each benchmark set and measured their minimum, mean and maximum values per benchmark set.

Results. The rel_flex_h results are grouped by benchmark set and their minimum, mean, and maximum per benchmark set are listed in Table 2.

Table 2. Statistics of flexibility ratio's rel_flex_h of decoupling updates vss static decoupling per benchmark set.

Benchmark set	Min	Mean	Max
bdh-agent-problems	1.00	1.00	1.002
Diamonds	1.34	1.95	2.39
Chordalgraphs	1.08	1.31	1.65
NeilYorkeExamples	1.20	1.74	2.39
sf-fixedNodes	1.21	1.38	1.78

As can be seen, except for bdh-agent-problems, dynamic updating of a temporal decoupling increases the mean flexibility per variable rather significantly: For example, in the diamonds and NeilYorke benchmark sets, updating almost doubled the flexibility per free time variable.[11]

We conclude that, based on this set of benchmarks, one might expect a significant increase in flexibility if a decoupling is updated after changes in commitments have been detected, compared to the case where no updating is provided.

To verify whether the heuristic was able to reduce the computation time significantly, unfortunately, we can only present partial results. The reason is that for the more difficult instances in these benchmark sets, computing the updates with an exact minimum matching method was simply too time-consuming.[12] We therefore selected from each benchmark set the easy[13] instances and collected them in the easy-<benchmark> variants of the original benchmark sets. We then measured the mean computation time per benchmark set for both the exact and heuristic updating method and also the mean performance ratio rel_flex/rel_flex_h of the exact versus the heuristic update method. The latter metric indicates how much the exact method outperforms the heuristic method in providing flexibility per time variable. The results are collected in Table 3.

[11] The reason that in the bdh-agent problems the updating did not increase, is probably due to the fact that in these instances, as we observed, the flexibility was concentrated in a very few time point variables. Eliminating the flexibilities by commitments of these variables did not affect the flexibility of the remaining variables that much.

[12] Some of the more difficult benchmark problems even did not finish within 36 h.

[13] A benchmark problem instance in [11] was considered to be "easy" if the exact update method finished in 15 min. For the bdh-agent set we collected the instances until easybdh 8_10_50_350_49, for the diamonds set all instances up to diamonds-38-5.0, for the chordal graphs all instances until chordal-fixedNodes-150,5,1072707262, for the NeilYorkeExamples all instances until ny-419,10, and for the sf-fixedNodes the instances until sf-fixedNodes-150,5,1072707262.

Table 3. Comparing the time (sec.) and performance ratio of the exact and heuristic updating methods (easy variants of benchmark instances)

Benchmark set	CPU heuristic	CPU exact	Performance ratio
easy-bdh-agent-problems	0.21	2.75	1.00
easy-diamonds	0.38	67.65	1.05
easy-chordalgraphs	0.78	12.6	1.06
easy-NeilYorkeExamples	0.61	135.62	1.01
easy-sf-fixedNodes	0.13	4.71	1.03

From these results we conclude that even for the easy cases, the heuristic clearly outperforms the exact update method, being more than 10 times and sometimes more than 200 times faster. We also observe that the heuristic method does not significantly lose on flexibility: The performance ratio's obtained are very close to 1.

7 Conclusions and Discussion

We have shown that adapting a decoupling after a new variable commitment has been made in most cases significantly increases the flexibility of the free, non-committed, variables. We also showed that this updating method did not induce any disadvantage to the actors involved: every commitment is preserved and the current decoupling bounds are never violated. We introduced a simple heuristic that replaces the rather costly computation of a decoupling with maximum flexibility by a computation of a decoupling with maximal flexibility. This heuristic reduces the computation time per update from $O(n^3)$ to $O(n \cdot \log n)$ (amortized) time. As we showed experimentally, the computational investments for the heuristic are significantly smaller, while we observed almost no loss of flexibility compared to the exact method.

In the future, we want to extend this work to computing flexible schedules for STPs: Note that the update heuristic also can be used to find a maximal flexible schedule given a solution s of an STP. Such a solution is nothing more than a non-optimal decoupling (s, s) for which, by applying our heuristic, an $O(n \log n)$ procedure exists to find an optimal flexible schedule. Furthermore, we are planning to construct dynamic decoupling methods in a distributed way, like existing approaches to static decoupling methods have done.

References

1. Boerkoel, J., Durfee, E.: Distributed reasoning for multiagent simple temporal problems. J. Artif. Intell. Res. (JAIR) **47**, 95–156 (2013)
2. Brambilla, A.: Artificial Intelligence in Space Systems: Coordination Through Problem Decoupling in Multi Agent Planning for Space Systems. Lambert Academic Publishing, Germany (2010)

3. Dechter, R.: Constraint Processing. Morgan Kaufmann, USA (2003)
4. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artif. Intell. **49**, 61–95 (1991)
5. Floyd, R.: Algorithm 97: shortest path. Commun. ACM **5**(6), 345 (1962)
6. Hunsberger, L.: Algorithms for a temporal decoupling problem in multi-agent planning. In: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI-02), pp. 468–475 (2002)
7. Hunsberger, L.: Group decision making and temporal reasoning. Ph.D. thesis, Harvard University, Cambridge, Massachusetts (2002)
8. van Leeuwen, P., Witteveen, C.: Temporal decoupling and determining resource needs of autonomous agents in the airport turnaround process. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2009, vol. 02, pp. 185–192. IEEE Computer Society, Washington, DC, USA (2009)
9. Mountakis, S., Klos, T., Witteveen, C.: Temporal flexibility revisited: maximizing flexibility by computing bipartite matchings. In: Proceedings Twenty-Fifth International Conference on Automated Planning and Scheduling (2015). http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10610
10. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall Inc., Upper Saddle River (1982)
11. Planken, L.: Algorithms for simple temporal reasoning. Ph.D. thesis, Delft University of Technology (2013)
12. Planken, L., de Weerdt, M., van der Krogt, R.: Computing all-pairs shortest paths by leveraging low treewidth. J. Artif. Intell. Res. **43**(1), 353–388 (2012). http://dl.acm.org/citation.cfm?id=2387915.2387925
13. Potra, F.A., Wright, S.J.: Interior-point methods. J. Comput. Appl. Math. **124**(1–2), 281–302 (2000). http://www.sciencedirect.com/science/article/pii/S0377042700004337. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations
14. Wilson, M., Klos, T., Witteveen, C., Huisman, B.: Flexibility and decoupling in the simple temporal problem. In: Rossi, F. (ed.) Proceedings International Joint Conference on Artificial Intelligence (IJCAI), pp. 2422–2428. AAAI Press, Menlo Park, CA (2013). http://ijcai.org/papers13/Papers/IJCAI13-356.pdf
15. Wilson, M., Klos, T., Witteveen, C., Huisman, B.: Flexibility and decoupling in simple temporal networks. Artif. Intell. **214**, 26–44 (2014)