Schriftenreihe Fachbereich Informatik

2018-1

Informatik
Hauptcampus

HOCHSCHULE TRIER

Uniform Decomposition of University Course Timetabling

Britta Herres[*] H. Schmitz[*†]

March 13, 2018

Abstract

Suppose we like to find non-overlapping periods for a set of events which may have multiple teachers assigned, is that easy or hard in terms of complexity? Or assume that only a single teacher is fixed per event, but we like to allocate rooms and periods simultaneously. What if a single teacher and a room is already given and we look for periods alone? And how do requests of teachers for specific rooms or period conflicts of students change the complexities of these questions from University Course Timetabling (UCT)?

We provide a complete hard/easy-list of all UCT subproblems derived from a typical set of hard constraints. We obtain this list with a systematic study of the fine structure of UCT in terms of embedded subproblems w.r.t. the order in which rooms, periods and teachers are assigned to events. This kind of subproblems appear in practice when some entities in a timetable are fixed while the assignments of others are (re-)computed, and they also appear as necessary conditions for the existence of feasible timetables. Moreover, we identify which of the seemingly different subproblems are essentially the same computational tasks by reducing them to the same bipartite assignment problem, and we discuss some variations of constraints.

1 Introduction

In University Course Timetabling (UCT) we usually try to allocate resources to teaching events such that the resulting timetable can be implemented throughout a teaching period without severe conflicts, or – even more – such that some notion of a 'good' timetable is met (for an overview see e.g. [McC06, Rud15, KS13, BKH15]). Due to the many variations of this setting in different institutions there is a high diversity within the family of UCT problems. On the other hand, there is also something like a core set of constraints that appear in many scenarios and that determine the minimum complexity we have to face at least when it comes to computations, e.g., rooms and teachers must not be assigned more than once in the same period. These hard constraints turn the general problem of allocating rooms, periods and sometimes also teachers and students into a hard computational task [EIS75, CK96].

We analyze in this paper the structure of UCT problems in terms of complexity based on a typical set of hard constraints. We gain this structure by looking at a natural notion of a *subproblem* where we consider the required entities (rooms, periods, teachers) separately and distinguish the order in which they are assigned to events. More precisely, we vary for each entity the cases whether this entity is not considered at all, or if a feasible assignment for this entity is already given in the input, or if a feasible assignment for this entity needs to be computed. We also distinguish single and multiple teachers per event and if event choices of

[*]Hochschule Trier, Schneidershof, D-54293 Trier, Germany
[†]Corresponding author: H.Schmitz@hochschule-trier.de

students are included or not. This gives a multitude of 66 subproblems that are embedded in this sense in the overall model, among them the opening questions from our abstract.

The role of subproblems in practical applications can be seen at least in the following ways. They can appear as *building blocks* in algorithms that go back and forth between single assignments, e.g., fix an assignment of periods, then assign teaching assistants if possible – if not, go back and assign (other) periods, and so on. If some assignments are fixed, then a subproblem can also be understood as a *recomputation problem* if parts of the basic data have changed, e.g., if teachers meanwhile have other availabilites. Finally, we can think of subproblems as *consistency checks* for the data provided since partial assignments constitute necessary conditions that are reasonable to check before a costly computation for the general problem is initiated. We think that it is fundamental to know which of these subproblems are (as) hard (as the general problem), and which of them are computationally easy. So our results also support approaches that are based on attacking UCT timetabling by distinguishing easy and hard portions of the problem – approaches that have been successful in the past, see e.g. [Kos05, LL12].

On the technical side, we cope with the multitude of subproblems in two ways. We derive from our set of hard constraints an order relation on subproblems, that is consistent with reductions and that we use to propagate complexity results among subproblems (Theorem 3.1). Secondly, we capture various subproblems at the same time by bipartite assignment problems (BAPs) with additional constraints and settle their complexity, inspired by [PZ80, TIR78]. The use of bipartite (multi-)graphs is a natural way to look at UCT problems, e.g., when computing feasible edge colorings [dW03,AdW02] or bipartite matchings [tEW01,Kin14,dW71]. We demonstrate that certain parameters of BAPs can make an important difference in terms of complexity, for example if conflicting sets form a partition or not. This is in line with related work where for particular settings also boundaries between hard/easy problem versions were identified, e.g. for the class-teacher problem [Got62], with respect to availability constraints [Csi65,EIS75] or students course choice [CK96]. Among others we show that a variation of the NP-complete problem known as Perfect Matching with Node Partitions [PZ80] can be solved efficienctly if we exploit the structure of the edge set as it appears in our UCT context (Prop. 4.4). Moreover, algorithms for BAPs can be used to solve all subproblems they capture – which in turn motivates to look at these BAPs more closely in future work.

The complexity results for all subproblems are summarized in Tables 3, 4 and 5 which can serve as quick look-up tables for hardness results if at least the constraints of our model (cf. Table 1) are present (e.g., the answers to the first three questions from the abstract are *hard*, *easy* and *hard*). The effect of adding constraints on teachers and rooms is given in Table 6. As a by-product we introduce a concise tuple notation for (sub-)problems, which can help to ease future discussions about the UCT problem family.

Notations. All variables like $i, j, k, l, \dots$ are nonnegative integers and all considered sets like $A, B, E, \dots$ are finite. With $\#A$ denote the cardinality of some set A. Initial segments of $\mathbb{N}^+$ are denoted by $[\alpha] = \{1, \dots, \alpha\}$. Finite families $\mathcal{A}$ of subsets over some universe A are written in calligraphic letters, as in $\mathcal{A} = \{A_i \subseteq A\}_{i \in [\alpha]}$. A partition $\mathcal{A}$ of a A is a set $\mathcal{A} = \{A_i \subseteq A\}_{i \in [\alpha]}$ of pairwise disjoint and non-empty blocks A_i such that $\bigcup_{i \in [\alpha]} A_i = A$. Unless stated otherwise, we consider undirected and simple bipartite graphs $G = (A \cup B, E)$ where each edge $ab \in E$ joins a vertex in A with a vertex in B. We call $G = (A \cup B, E)$ complete if $E = A \times B$. For a bipartite graph $G = (A \cup B, E)$ we say $M \subseteq E$ is a matching of A if $\#M = \#A$ and if for all distinct edges $ab, a'b' \in M$ it holds that $a \neq a'$ and $b \neq b'$. We will frequently make use of the term *assignment of A* where we only require that M has cardinality $\#A$ and that $a \neq a'$ for each pair of distinct $ab, a'b' \in M$. When we classify decision problems as easy or hard we do so in terms of computational complexity and mean that a problem $\mathcal{P}$ belongs to the class P or is shown to be NP-complete. With $\leq_m^p$ denote polynomial-time many-one reductions between decision problems. For definitions on complexity classes and reductions we refer to standard textbooks, e.g. [GJ90].

Our Model. We assume in our UCT model that a set E of events is given and we need to assign to each event e the following entities:

- one room r_e from a given set of rooms R, and
- one period p_e from a set of non-overlapping periods P, and
- either a single teacher t_e or a set T_e of teachers, from a given set T.

We distinguish the cases of a single teacher and a set of teachers in order to discuss the implications later. In the latter case the number $\#T_e$ of needed teachers is specified per event and is at least one. There is also a set S of students given who already have selected their events, which corresponds to a simple course model where students choice is conducted on event level. An alternative way to look at it is that students have already been scheduled to certain events in the input, i.e., the pre-timetabling setting of student sectioning. Note that this can also be used to model curriculum-based timetabling where single students represent their whole curriculum. We refrain here from including other forms of student sectioning in the same model and refer to [DPS16, Sch16] for complexity results for student sectioning.

A timetable is feasible for the given instance if assignments for all events can be found such that specified constraints are satisfied. We only consider so-called hard constraints required for a practical implementation of a timetable, e.g., two different events must not be scheduled in the same room during the same period. The list of constraints for our model is given in Table 1. Binary constraints refer to all pairs of given events while unary constraints restrict possible assignments for each single event. On one hand, this list is not minimal in the sense that, e.g., for some events a slight violation of room capacities might be acceptable or even necessary in practice. On the other hand, due to the many variations of problems from the UCT family it is clear that additional hard constraints could be needed which add to the complexity of (sub-)problems, e.g., if one event has to follow another event in the next period [tEW01]. For our structural analysis we stick here to this list of *typical* constraints that appear in many scenarios.

Table 1: Hard constraints considered in our model.

	Constraint	Type	Comment
C_1	room conflict	binary	a room can be scheduled at most once a period
C_2	teacher conflict	binary	a teacher can be scheduled at most once a period
C_3	student conflict	binary	a student can be scheduled at most once a period
C_4	event availability	unary	an event must be scheduled at its available periods
C_5	teacher ability	unary	a teacher must be capable to teach this event
C_6	room capacity	unary	a room's capacity must not be exceeded
C_7	room availability	unary	a room must be scheduled at its avail. periods
C_8	teacher availability	unary	a teacher must be scheduled at his avail. periods
C_9	number of teachers	unary	each events has the number of needed teachers

Although completeness of assignments is implicitly understood, i.e., for all events we have to assign exactly one room, one period and the needed teachers, we list constraint C_9 to emphasize that a single teacher is possibly not enough. Also note that there is no constraint that refers to the course choice of students explicitely, but it needs to be respected during period and room assignments (C_3, C_6). All problems we consider are decision problems where we ask if a feasible timetable exists w.r.t. the input data and the set of all active constraints (in the next section we define subproblems for which only a subset of the constraints from Table 1 is reasonable to consider). Observe that the corresponding optimization problems asking to maximize the number of events in a feasible assignment are not easier in term of complexity.

The rest of the paper is organized as follows. In the next section we define all subproblems that result from our decomposition approach, introduce a concise notation for them and explain their role in the UCT setting. Then we show in Section 3 how subproblems are related in terms of polynomial-time many-one reductions. In Section 4 we introduce bipartite assignment problems with additional constraints and classify them as easy/hard. We use these general results to determine the complexity of all our subproblems in Section 5. In Section 6 we look more detailed at those of them that are sensitive to slight changes of constraints concerning rooms and teachers. Finally, we conclude with notes on future work in Section 7.

2 Decomposition and the Role of Subproblems

For each of the assignments of rooms, periods and teachers to events we distinguish in subproblems the cases

(1) whether this entitiy is not considered at all, or

(2) if a feasible assignment for this entity is already given in the input, or

(3) if a feasible assignment for this entity needs to be computed.

Additionally, we like to vary the cases of single/multiple teachers and if event choices of students are included or not.

Subproblem notations. In our tuple notation for subproblems we write '$-$' if an entitiy is not taken into account (1), we indicate a given assignments by italic letters (2), and assignments that need to be computed by bold letters (3). The first component refers to the assigment of rooms ($-$, r or $\mathbf{r}$), the second to the assignment of periods ($-$, p or $\mathbf{p}$), the third to single or multiple teachers ($-$, t, T, $\mathbf{t}$ or $\mathbf{T}$), and the last component informs whether the given course choices of students are included ($-$ or S). As an example consider the problem $(\mathbf{r}, \mathbf{p}, t, S)$ where one teacher for each event is already fixed in a feasible way (i.e., teacher abilities are respected, see C_5) and event choices of students need to be respected. Then the problem asks to find a feasible assignment of a room and a period for all events, according to the remaining constraints and while leaving the given teacher assignment unchanged. Note that '$-$' in the last component would exclude checking room capacities and period clashes for students (C_6, C_3).

Each component in this notation can be chosen independently giving 66 many problems where at least one component is set in bold letters. There are 40 problems which ask for the assignment of exactly one type of entity, and 26 multi-assignment problems. When we use the term *subproblem* we mean a decision problem defined by any of these 66 notations. All subproblems clearly belong to the class NP since timetables are polynomially length-bounded w.r.t. input length and all constraints can be verified in polynomial-time for a given timetable. For ease of notations we write '$*$' in a component if the assignment might or might not be given in the input, and talk about the respective subproblems simultaneously, e.g., $(\mathbf{r}, *, t, S)$ refers to $(\mathbf{r}, -, t, S)$ and $(\mathbf{r}, p, t, S)$.

Active constraints. It needs to be clear from the problem notation what constraints from Table 1 apply to each subproblem: The general policy is that we consider *all* constraints that can be reasonably applied, i.e., all constraints for which all required entities are present in the subproblem notation, either as part of the input or part of the output. This constraint set can be easily determined for each subproblem by Table 2 which relates constraints and required entities. If the assignment given in some row needs to be computed, then the constraints in the second column are obligatory while the other constraints in that row apply if the component in the column heading is also set. If more than one assignment needs to be computed, then constraint sets are joined. As an example consider $(-, \mathbf{p}, \mathbf{t}, S)$. We need to consider constraints $\{C_4\} \cup \{C_2, C_8\} \cup \{C_3\}$ from row $\mathbf{p}$ since $\mathbf{t}$ and S are mentioned in the problem notation, additional we have constraints $\{C_5, C_9\} \cup \{C_2, C_8\} \cup \emptyset$ from row $\mathbf{t}$ due to $\mathbf{p}$ and S. So together we have C_2 (period clashes for the assigned teachers must be avoided), C_3 (period clashes for the given students must be avoided), C_4 (events must be available at the period to assign), C_5 (teachers must have the ability to teach the events they are assigned to), C_8 (assigned teachers must be available at the assigned period) and C_9 (the required number of teachers have to be assigned).

Table 2: Constraint sets for subproblems. Each row represents the constraints which must be satisfied if we ask for this assignment and if the column heading appears in the problem notation.

to be computed	obligatory	r or $\mathbf{r}$	p or $\mathbf{p}$	t/T or $\mathbf{t}/\mathbf{T}$	S
$\mathbf{r}$	$\emptyset$	–	$\{C_1, C_7\}$	$\emptyset$	$\{C_6\}$
$\mathbf{p}$	$\{C_4\}$	$\{C_1, C_7\}$	–	$\{C_2, C_8\}$	$\{C_3\}$
$\mathbf{t}/\mathbf{T}$	$\{C_5, C_9\}$	$\emptyset$	$\{C_2, C_8\}$	–	$\emptyset$

Composition, recomputation and consistency checks. Suppose we want to compute assignments for two or more entities simultaneously. Clearly, if there is no feasible assignment for a single entity alone, then there is also no feasible assignment for this entity if we additionally ask for other assignments at the same time. But if the answer is positive, one can think of composing the computations for subproblems by taking the assignment for the first entity as fixed input to the second computation, and so on (although we consider decision problems, any reasonable algorithm will actually compute a solution if the answer is positive). Note that in general this straight-on composition of single-assignment problems does not provide a decision algorithm for the respective multiple-assignment problem since each subproblem asks for the existence of an *arbitrary* feasible assignment. So if a *particular* assignment for some entity is fixed beforehand, there might not exist a feasible assignment for another entity under this restriction – although simultaneous assignments for both entities are possible. However, this still leaves room to construct algorithms that go back and forth between subproblems in various ways. E.g., in order to find a solution to $(\mathbf{r}, \mathbf{p}, t, -)$ one could start with $(-, \mathbf{p}, t, -)$ to obtain a period assignment p, then try to solve $(\mathbf{r}, p, t, -)$ and if this fails, go back to $(-, \mathbf{p}, t, -)$ for a different p'. We think it is important to know which of these subproblems can be solved efficiently, and which are computationally hard, as indicated by our hard/easy classification of all subproblems below.

Subproblems with fixed assignments in the input can also be understood as recomputation problems if parts of the basic data have changed. As another example suppose we already have a solution to $(\mathbf{r}, \mathbf{p}, \mathbf{T}, S)$ and teacher availabilities have changed. Then $(r, p, \mathbf{T}, S)$ asks for a (new) feasible assignment of teachers to events while keeping the previous room and period assignments, whereas $(\mathbf{r}, \mathbf{p}, T, S)$ keeps the allocation of teachers to events but re-schedules rooms and periods. Observe that there are six subproblems that are reasonable to consider as recomputations in order to react to the change of data besides computing a completely new solution to $(\mathbf{r}, \mathbf{p}, \mathbf{T}, S)$. Again, we think that it is fundamental to know which of these problems can be solved efficiently and which are as hard as the general problem.

Finally, we think of subproblems as consistency checks for the data provided. Suppose we look for a solution to $(\mathbf{r}, \mathbf{p}, \mathbf{t}, S)$. Before we initiate a costly computation for the general problem, it is reasonable to check the necessary condition that solutions to $(\mathbf{r}, -, -, S)$, to $(-, \mathbf{p}, -, S)$ and to $(-, -, \mathbf{t}, S)$ must exist, even more if these checks are easy to carry out. Note that positive answers to the remaining double-assignment problems constitute necessary conditions as well. To sum up, subproblems play an important role as a basis for understanding the structure of a given specific UCT scenario.

3 Reductions between Subproblems

Intuitively, a subproblem with a fixed assignment for some entity should not be easier than without considering this entity at all, and a subproblem where we need to compute an assignment should not be easier than the case when a feasible assignment is already given in the input for that entity. It turns out that we can make this intuition precise in terms of polynomial-time many-one reductions between subproblems, but we will also find out that some details are quite sensitive to the actual set of constraints. To formalize this we start with the definition of a partial order $\prec$ on (the symbols in) each component of our subproblem notation given by the relations depicted in Fig. 1. We denote by $\preceq$ the reflexive and transitive closure of $\prec$ and extend it in a natural way to a product order on subproblem notations. So for example it holds that $(-, \mathbf{p}, t, -) \preceq (r, \mathbf{p}, \mathbf{T}, S)$ whereas $(-, \mathbf{p}, -, S)$ and $(\mathbf{r}, p, T, -)$ are incomparable. We use the following theorem to propagate complexity results among subproblems.

$$\begin{tikzcd}
 & & & \mathbf{p} & \mathbf{t} \arrow[r] & \mathbf{T} & \\
r & & \mathbf{r} & p \arrow[u] & t \arrow[u] \arrow[r] & T \arrow[u] & S \\
 & - \arrow[ul] \arrow[ur] & & - \arrow[u] & - \arrow[u] & & - \arrow[u]
\end{tikzcd}$$

Figure 1: Partial order on (the symbols in) each component of the subproblem notation.

Theorem 3.1 *If $\mathcal{P}$ and $\mathcal{P}'$ are subproblems, then $\mathcal{P} \preceq \mathcal{P}'$ implies $\mathcal{P} \leq_m^p \mathcal{P}'$.*

Proof We consider each component and each relation from Fig. 1 separately due to transitivity of $\leq_m^p$. Let x be an instance of $\mathcal{P}$. We show how to construct an instance $f(x)$ of $\mathcal{P}'$ such that x is a yes-instance for $\mathcal{P}$ if and only if $f(x)$ is a yes-instance for $\mathcal{P}'$. It will be obvious from the instructions for $f(x)$ that its construction can be carried out in time polynomial in the length of x.

1. Room component: Let $\mathcal{P} = (-, b, c, d)$ and $\mathcal{P}' = (r, b, c, d)$ for fixed $b \in \{-, p, \mathbf{p}\}$, $c \in \{-, t, T, \mathbf{t}, \mathbf{T}\}$ and $d \in \{-, S\}$. Moreover, let $C \subseteq \{C_2, C_3, C_4, C_5, C_8, C_9\}$ be the set of constraints for $\mathcal{P}$ as derived from $(-, b, c, d)$. We need to add a feasible room assignment to x such that C_1 and C_7 hold if $b \neq -$, and such that C_6 holds if $d \neq -$. So we introduce a synthetic set of rooms R that are always available and that have sufficient capacities for all

students. Then we fix in $f(x)$ for each event one room from R such that all pairs of events have different rooms assigned, and provide this room assignment as an additional input. Note that these rooms are feasible w.r.t. C_1 and C_7 no matter how periods are assigned to events, and that it is feasible w.r.t. C_6 no matter how many students have selected an event. Now suppose that x is a yes-instance of $\mathcal{P}$ and let tt be a witnessing timetable that satisfies all constraints in C, and let tt' be the same as tt but with the above room assignment added. Then tt' does not change the room assignment fixed in $f(x)$ and it still satisfies all constraints in C because no $C_i \in C$ refers to rooms. So $f(x)$ is a yes-instance of $\mathcal{P}'$ as witnessed by tt'. Conversely, if tt' is a timetable witnessing that $f(x)$ is a yes-instance of $\mathcal{P}'$, then dropping all room assignments gives a feasible timetable for input x and constraint set C.

For the second case keep all notations but let $\mathcal{P}' = (\mathbf{r}, b, c, d)$. Then we just add the set of rooms R to obtain $f(x)$ from x which allows for feasible room assignments without affecting the other constraints. So if x is a yes-instance of $\mathcal{P}$, then there is also a timetable witnessing that $f(x)$ is a yes-instance of $\mathcal{P}'$ since an additional feasible room assignment exists, and dropping such a room assignment conversely gives a feasible timetable for x.

2. Period component: Let $\mathcal{P} = (a, -, c, d)$ and $\mathcal{P}' = (a, p, c, d)$ for fixed $a \in \{-, r, \mathbf{r}\}$, $c \in \{-, t, T, \mathbf{t}, \mathbf{T}\}$ and $d \in \{-, S\}$. Moreover, let $C \subseteq \{C_5, C_6, C_9\}$ be the set of constraints for $\mathcal{P}$. We proceed as before and fix a period assignment in advance that does not reduce the set of feasible timetables in $\mathcal{P}'$ although more constraints need to be respected. So we introduce a synthetic set of periods P and pre-assign each event to a different period from P. Additionally, we set the data in $f(x)$ such that all events, all rooms and also all teachers are always available. The pre-assignment of periods is feasible since there are no period clashes for rooms, teachers and students (C_1, C_2, C_3) and all availablities are vacuously fulfilled (C_4, C_7, C_8). So solutions for x may serve as solutions for $f(x)$ and vice versa.

For the second case let $\mathcal{P} = (a, p, c, d)$ and $\mathcal{P}' = (a, \mathbf{p}, c, d)$ and observe that both subproblems have the same set of constraints attached. However, in x there is an assignment of periods to events already fixed and we have to ensure that this given assignment is the only one that can be computed for instance $f(x)$ – without actually providing it in the input. This can be achieved by altering the availablities of events such that each event is only available at the periods assigned in x. If we leave all other data unchanged, then the set of feasible timetables for x and $f(x)$ coincide.

3. Teacher component: Reductions for this component can be easily obtained from the following observations: To proceed in the third component from $-$ to t we can introduce as before a synthetic set T of teachers that are always available, and assign to each event a different teacher. If a teacher assignment is already given in the input we can reduce to $\mathbf{t}$ or $\mathbf{T}$, respectively, if we use the abilities of teachers per event to allow that only the formerly fixed assignment remains feasible. Finally, observe that the single teacher requirement is just a special case of multiple teachers, and we can set $\#T_e = 1$ when reducing from $\mathbf{t}$ to $\mathbf{T}$.

4. Student component: It remains to look at $\mathcal{P} = (a, b, c, -)$ and $\mathcal{P}' = (a, b, c, S)$ where we need to adapt x such that the additional constraints C_3 and C_6 do not cut the set of feasible timetables. To do so we introduce a synthetic set S of students each of which has selected a single but different event, and all rooms have capacitiy at least 1 (note that we could set $S = \emptyset$ as well). $\square$

The proof shows why the partial order from Fig. 1 looks as it does when we stick to the constraints from Table 1: Without any other constraints we don't know how to fix a given room assignment when we are asked to compute one, in contrast to periods and teachers where we can use availabilities of events and the abilities of teachers for this purpose. However, if we add a constraint like the requirement of particular resources of rooms that are needed for an event, then we also have $- \to r \to \mathbf{r}$ in the $\prec$-order for the room component. So if constraints are added or removed, the diagram in Fig. 1 changes accordingly.

4 Bipartite Assignment Problems with Additional Constraints

In this section we introduce rather abstract problem definitions and determine their complexity. We do so in a uniform way based on assignments in bipartite graphs $G = (A \cup B, E)$ – a natural way to look at UCT subproblems. When we apply these problems in our context, then A will mostly be a set of events. The most basic version only requires that for each event exactly one assignment is possible.

Basic Bipartite Assignment Problem (BAP)
Input: A bipartite graph $G = (A \cup B, E)$.
Question: Is there an assignment $M \subseteq E$ for A?

Obviously, a feasible assignment can be found iff each $a \in A$ has at least degree one, which we may assume w.l.o.g. since events with no possible assignments at all can be initially discarded. So BAP without any other constraints is a trivial problem. Next we turn to the case where different events must not be mapped to the same entity on the right-hand side.

BAP($\mathcal{A}$)
Input: A bipartite graph $G = (A \cup B, E)$ and a family $\mathcal{A} = \{A_i \subseteq A\}_{i\in[\alpha]}$ of conflicting sets.
Question: Is there an assignment $M \subseteq E$ for A such that for all distinct ab, $a'b' \in M$ it holds that $b \neq b'$ whenever $a, a' \in A_i$ for some i?

As an example consider $(-, \mathbf{p}, T, -)$ where events that share at least one teacher need to be scheduled in different periods, which can be modeled by conflicting sets A_t for each teacher. We notice that different properties of $\mathcal{A}$ are crucial for the complexity of BAP($\mathcal{A}$).

Proposition 4.1

1. BAP($\mathcal{A}$) *is NP-complete, even in the restricted case that* G *is complete and that* $\#\{i|a \in A_i\} = 2$ *for all* $a \in A$.
2. BAP($\mathcal{A}$) *is in P in the resctricted case that* $\mathcal{A}$ *is a partition of* A.

Proof 1. Clearly BAP($\mathcal{A}$) is an NP problem. We show completeness by reduction from EDGE COLORING for arbitrary simple graphs [Hol81]. Let $G = (V, E)$ and some integer k be given, and we ask if we can assign one out of k colors to each edge such that no adjacent edges share the same color. We construct a complete bipartite graph with vertex partitions $A = E$ and $B = [k]$ and obtain $G' = (E \cup [k], E \times [k])$. Moreover, we set $\mathcal{A} = \{A_v \mid v \in V\}$ where A_v is the set of all edges incident to v and hence conflicting w.r.t. color. Note that each edge appears exactly twice in some set A_v. Then (G, k) is a yes-instance of EDGE COLORING if and only if $(G', \mathcal{A})$ is a yes-instance of BAP($\mathcal{A}$).

2. If $\mathcal{A}$ is a partition of A, we look at each block $A_i \subseteq A$ separately and need to find a perfect matching in all of the hereby restricted bipartite graphs [HK73]. □

We also note that the first statement also holds in the restricted case that G is complete and that $\#A_i = 2$ for all i if we reduce from VERTEX COLORING [Kar72] and swap edges and vertices in the reduction.

For the next problem observe that with a single family $\mathcal{A}$ of conflicting sets we are only capable of modeling conflicts of the same kind, e.g., caused by given teacher assignments. If more kinds of conflicts are present, e.g., if also room assignments already exist, then we need to extend our definitions to capture these constraints as well.

BAP($\mathcal{A}$,$\mathcal{A}'$)

Input: A bipartite graph $G = (A \cup B, E)$ and families $\mathcal{A} = \{A_i \subseteq A\}_{i\in[\alpha]}$ and $\mathcal{A}' = \{A'_j \subseteq A\}_{j\in[\alpha']}$ of conflicting sets.

Question: Is there an assignment $M \subseteq E$ for A such that for all distinct ab, $a'b' \in M$ it holds that $b \neq b'$ if $a, a' \in A_i$ or $a, a' \in A'_j$ for some i, j?

It is immediately clear from Prop. 4.1.1 that BAP($\mathcal{A}$,$\mathcal{A}'$) is NP-complete even if G is complete. So we only look at the case when $\mathcal{A}$ and $\mathcal{A}'$ are both partitions of A.

Proposition 4.2 *Assume that $\mathcal{A}$ and $\mathcal{A}'$ are partitions of A.*

1. BAP($\mathcal{A}$,$\mathcal{A}'$) *is NP-complete, even if* $\#(A_i \cap A'_j) \leq 1$ *for all* A_i, A'_j.
2. BAP($\mathcal{A}$,$\mathcal{A}'$) *is in P in the restricted case that G is complete.*

Proof 1. Clearly BAP($\mathcal{A}$,$\mathcal{A}'$) is an NP problem. We show completeness by reduction from LIST EDGE COLORING for bipartite graphs [Mar05]. Let $G = (X \cup Y, E)$ and lists $L(e)$ of colors for each edge $e \in E$ be given, and we ask if we can assign one of the colors in $L(e)$ to each edge e such that no adjacent edges share the same color. We construct a bipartite graph G' with vertex partitions $A = E$ and $B = \bigcup_{e\in E} L(e)$ and include edges $ec \in A \times B$ if $c \in L(e)$. Moreover, we set $\mathcal{A} = \{A_x \mid x \in X\}$ and $\mathcal{A}' = \{A'_y \mid y \in Y\}$ where A_x and A'_y are the sets of all edges incident to $x \in X$ and $y \in Y$, respectively. Observe that $\mathcal{A}$ and $\mathcal{A}'$ are both partitions of A, and that $\#(A_x \cap A'_y) \leq 1$ since otherwise there is more than one edge joining x, y in G. Then $(G, \{L(e)\}_{e\in E})$ is a yes-instance of LIST EDGE COLORING if and only if $(G', \mathcal{A}, \mathcal{A}')$ is a yes-instance of BAP($\mathcal{A}$,$\mathcal{A}'$).

2. We show this by reduction to EDGE COLORING for bipartite multigraphs [COS01]. So let $G = (A \cup B, A \times B)$ and partitions $\mathcal{A}$, $\mathcal{A}'$ of A be given. We construct $G' = ([\alpha] \cup [\alpha'], E)$ as a bipartite multigraph with l edges between some $i \in [\alpha]$ and $j \in [\alpha']$ if $\#(A_i \cap A'{}_j) = l$, and ask if G' can be properly colored with $k = \#B$ many colors. Then $(G, \mathcal{A}, \mathcal{A}')$ is a yes-instance of BAP($\mathcal{A}$,$\mathcal{A}'$) if and only if (G', k) is a yes-instance of EDGE COLORING for bipartite multigraphs. □

We summarize the complexity results by now for later reference and distinguish different parameter combinations.

Ref.	Problem	$\#\{i \mid a \in A_i\}$	α	$\#\{i \mid a \in A'_i\}$	α'	complete?	P/NPC	comment
①	BAP	-	-	-	-	no	P	
②	BAP($\mathcal{A}$)	$= 1$	arb.	-	-	no	P	Prop. 4.1.2
③	BAP($\mathcal{A}$)	$= 2$	arb.	-	-	yes	NPC	Prop. 4.1.1
④	BAP($\mathcal{A}$)	arb.	arb.	-	-	no	NPC	unrestricted
⑤	BAP($\mathcal{A}$,$\mathcal{A}'$)	$= 1$	arb.	$= 1$	arb.	yes	P	Prop. 4.2.2
⑥	BAP($\mathcal{A}$,$\mathcal{A}'$)	$= 1$	arb.	$= 1$	arb.	no	NPC	Prop. 4.2.1
⑦	BAP($\mathcal{A}$,$\mathcal{A}'$)	arb.	arb.	arb.	arb.	no	NPC	unrestricted

In order to model cases of two simultaneous assignments as in (**r**,p,**t**,-) we further extend our constraints and consider two families of conflicting sets also on the right-hand side for vertices in B.

BAP($\mathcal{A}$;$\mathcal{B}$,$\mathcal{B}'$)

Input: A bipartite graph $G = (A \cup B, E)$ and families $\mathcal{A} = \{A_i \subseteq A\}_{i \in [\alpha]}$, $\mathcal{B} = \{B_j \subseteq B\}_{j \in [\beta]}$ and $\mathcal{B}' = \{B'_k \subseteq B\}_{k \in [\beta']}$ of conflicting sets.

Question: Is there an assignment $M \subseteq E$ for A such that for all distinct ab, $a'b' \in M$ it holds that $\{b, b'\} \not\subseteq B_j$ for no j and $\{b, b'\} \not\subseteq B'_k$ for no k whenever $a, a' \in A_i$ for some i?

Observe that if we set $\mathcal{B} = \{\{b\} | b \in B\}$ as singeltons and omit $\mathcal{B}'$ then we meet the requirements of BAP($\mathcal{A}$), hence we know again from Prop. 4.1.1 that BAP($\mathcal{A}$;$\mathcal{B}$,$\mathcal{B}'$) is NP-complete even if G is complete. However, in our UCT context it might be that $\mathcal{B}$ and $\mathcal{B}'$ are partitions of B.

Proposition 4.3 *Assume $\mathcal{B}$ and $\mathcal{B}'$ are partitions of B.*

1. BAP($\mathcal{A}$;$\mathcal{B}$,$\mathcal{B}'$) *is NP-complete even if $\mathcal{A} = \{A\}$.*
2. BAP($\mathcal{A}$;$\mathcal{B}$,$\mathcal{B}'$) *is in P if $\mathcal{A} = \{A\}$ and if $\mathcal{B}$ or $\mathcal{B}'$ is a family of singletons.*

Proof 1. Clearly BAP($\mathcal{A}$;$\mathcal{B}$,$\mathcal{B}'$) is an NP problem. We show completeness by reduction from 3-Dimensional Matching (3DM) [Kar72]. Given finite and pairwise disjoint sets X, Y, Z with $\#X = \#Y = \#Z$ and some $J \subseteq X \times Y \times Z$ we need to decide whether a perfect matching $M \subseteq J$ exists. To do so, we set $A = X$ and $B = \{yz \in Y \times Z \mid \exists_x\ xyz \in J\}$ in our bipartite graph G. Edges $x(yz) \in A \times B$ exist iff $(x, y, z) \in J$. Partitions $\mathcal{B}$ and $\mathcal{B}'$ of B are induced by Y and Z, i.e., each B_y (B'_z) consists of all vertices from B containing y (z, respectively), which ensures that each of these vertices is chosen at most once. Note that we get a perfect matching since we ask for an assignment for A. Finally let $\mathcal{A} = \{A\}$ such that constraints need to be respected for all pairs of vertices from $A = X$. Then (X, Y, Z) is a yes-instance of 3DM if and only if $(G, \mathcal{A}, \mathcal{B}, \mathcal{B}')$ is a yes-instance of BAP($\mathcal{A}$;$\mathcal{B}$,$\mathcal{B}'$).

2. Now suppose that additionally and w.l.o.g. $\mathcal{B} = \{\{b\} | b \in B\}$ Then we only need to decide whether there is a matching of A in G. If partition $\mathcal{B}'$ is not a family of singeltons, then we merge all vertices from each $B'_k \in \mathcal{B}'$ into a single vertex and again ask for a matching of A. $\Box$

Finally we recall a variation of the NP-complete problem known as Perfect Matching with Node Partitions [PZ80] where M only has to be a matching of A instead of being perfect.

Bipartite Matching Problem with Vertex Partitions (BMP($\mathcal{A}$;$\mathcal{B}$))

Input: A bipartite graph $G = (A \cup B, E)$ and partitions $\mathcal{A}$ of A and $\mathcal{B}$ of B of conflicting sets.

Question: Is there a matching $M \subseteq E$ of A such that for all distinct ab, $a'b' \in M$ it holds that neither a, a' are in the same block of $\mathcal{A}$, nor b, b' are in the same block of $\mathcal{B}$?

As an example consider (**r**,**p**,t,-) where partitions $\mathcal{A}$ and $\mathcal{B}$ naturally appear when events are grouped by teachers, and rooms by available periods, resp.. However, in this case the edges between two blocks have a special structure due to the fact that events and teacher have no room restrictions in our model (note that the fourth component is $-$, we discuss additional constraints in Sec. 6). We make this observation precise with the following notations.

For $G = (A \cup B, E)$ we say that $E' \subseteq E$ is closed under product if for all distinct edges $ab, a'b' \in E'$ it holds that $ab', a'b \in E'$. A subgraph $G[A' \cup B']$ of G induced by $A' \subseteq A$ and $B' \subseteq B$ is closed under product if its edge set has this property. Finally, a bipartite graph G with partitions $\mathcal{A}$ and $\mathcal{B}$ is closed under product if each subgraph $G[A_i \cup B_j]$ for all i, j has this property.

Proposition 4.4 BMP($\mathcal{A}$;$\mathcal{B}$) *is in P if G is closed unter product w.r.t. $\mathcal{A}, \mathcal{B}$.*

Proof We show this by reduction to Max Flow [FF87]. Let $(G = (A \cup B, E), \mathcal{A}, \mathcal{B})$ be an instance of this problem, and call a block B_j connected to some $a \in A_i$ if there exists at least one edge ab with $b \in B_j$ (and vice versa).

Construction. We define a flow network $F = (V, E')$ as follows (cf. Figure 2): All edges in F have capacity one. The source s has an edge to all vertices of the first layer A. The second layer consists of all vertices $ij \in [\#\mathcal{A}] \times [\#\mathcal{B}]$ and $a \in A_i$ has an edge to some ij if it is connected to B_j. The third layer has a copy of each ij linked with a single edge. The last layer consist of all vertices from B. Each ij from the third layer has an edge to some $b \in B_j$ if b is connected to A_i. Finally, all $b \in B$ have an edge to the sink t. We ask for a flow f in F with value $v(f) = \#A$.

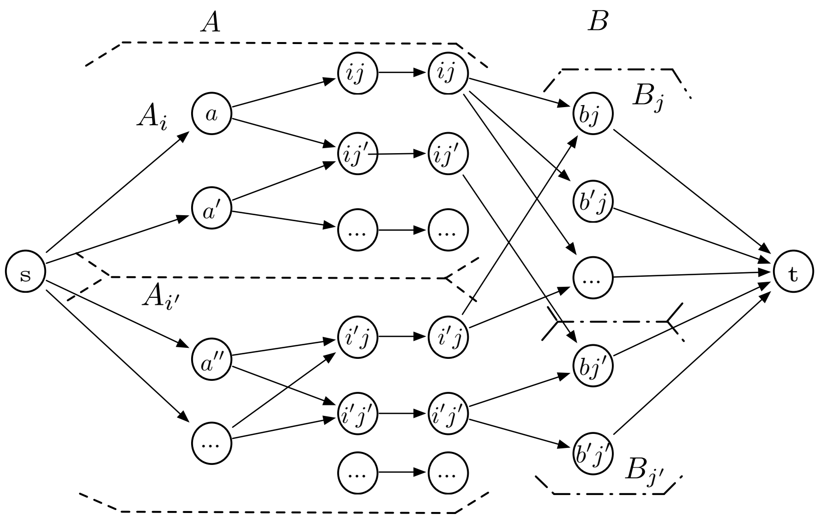

Figure 2: Sketch of a resulting flow network.

Assume that M is a matching of A such that for all distinct $ab, a'b' \in M$ in holds that neither a, a' are in the same block of $\mathcal{A}$ nor b, b' are in the same block of $\mathcal{B}$. For each $ab \in M$ with $a \in A_i$ and $b \in B_j$ we obtain disjoint s-t-paths each with one unit of flow on edges $(s, a), (a, ij), (ij, ij), (ij, b), (b, t)$. Note that this also works if G is not closed under product.

Conversely, suppose there is a flow f with value $v = \#A$. Then we define M as the set of all edges ab if the path $(a, ij), (ij, ij), (ij, b)$ carries one unit of flow. By construction it holds that a is connected to B_j and b is connected to A_i, so ab exists in G since it is closed under product. Edge set M has cardinality $\#A$ since each a has capacity one on the incoming edge. Moreover, all vertices b are pairwise distinct due to capacities on outgoing edges, hence M is a matching of A. There are no two edges in M connecting the same pair of blocks A_i, B_j due to capacity one on (ij, ij), and for each $a \in A_i$ and $b \in B_j$ there is at most one a-b-path in F. □

We summarize the remaining problems as follows.

Ref.	Problem	$\mathcal{A}$	$\mathcal{B}$ or $\mathcal{B}'$	P/NPC	comment
⑧	BAP($\mathcal{A}$;$\mathcal{B}$,$\mathcal{B}'$)	$\{A\}$	singletons	P	Prop. 4.3.2
⑨	BAP($\mathcal{A}$;$\mathcal{B}$,$\mathcal{B}'$)	$\{A\}$	arb. part.	NPC	Prop. 4.3.1
⑩	BAP($\mathcal{A}$;$\mathcal{B}$,$\mathcal{B}'$)	arb. part.	arb. part.	NPC	unrestr.
⑪	BMP($\mathcal{A}$;$\mathcal{B}$) with product closure	arb. part.	arb. part.	P	Prop. 4.4
⑫	BMP($\mathcal{A}$;$\mathcal{B}$)	arb. part.	arb. part.	NPC	[PZ80]
⑬	BMP($\mathcal{A}$;$\mathcal{B}$)	arb.	arb.	NPC	unrestr.

5 Complexities of all UCT Subproblems

In this section we derive the complexities of all UCT subproblems. In light of Theorem 3.1 it is sufficient for a complete classification to identify $\preceq$-maximal problems that are in P and $\preceq$-minimal problems that are NP-complete.

Theorem 5.1
Subproblems $(r, \boldsymbol{p}, -, -)$, $(r, p, \boldsymbol{T}, S)$, $(\boldsymbol{r}, \boldsymbol{p}, t, -)$, $(\boldsymbol{r}, p, \boldsymbol{T}, S)$ and $(-, \boldsymbol{p}, \boldsymbol{t}, -)$ are in P. Subproblems $(-, \boldsymbol{p}, T, -)$, $(-, \boldsymbol{p}, -, S)$ and $(r, \boldsymbol{p}, t, -)$ are NP-complete.

Proof For each problem we mention the respective reduction and provide some indication how the reduction works. We use problem references ⓘ from Section 4 and write →ⓘ if the current subproblem is reducible to problem ⓘ and ⓘ→ if problem ⓘ is reducible to our subproblem.

$(r, \mathbf{p}, -, -)$: →②
Events are grouped subject to common rooms into blocks A_i. Edges are present due to availabilities of events and rooms.

$(r, p, \mathbf{T}, S)$: →②
We partition events in $\mathcal{A}$ according to common periods. To ensure exactly $\#T_e$ distinct assignments of teachers we copy each event and the incident edges $\#T_e$ times. Note that all copies are in the same block of $\mathcal{A}$. Teacher abilities are respected by (non-)existence of edges.

$(\mathbf{r}, \mathbf{p}, t, -)$: →⑪
We partition events in $\mathcal{A}$ according to common teachers. All eligible room-period combinations are partitioned in $\mathcal{B}$ according to common periods. Edges are drawn between some event and a room-period combination if an event and its preassigned teacher are available in this period. Note that if an event is available at some period then the event is connected to all rooms in this period. Then the resulting instance BMP($\mathcal{A}$;$\mathcal{B}$) is closed under product.

$(\mathbf{r}, p, \mathbf{T}, S)$: →②
Assignments for rooms and teachers can be found independently in our setting since they do not share any constraints. Observe that if a period assignment is already present that is feasible for teachers and students, then we find feasible rooms when events are paritioned in $\mathcal{A}$ according to common periods. Edges exist if the room is eligible w.r.t. period availability and sufficient capacity for the given students. If a period assignment is already present that is feasible for rooms and students, then we can assign teachers as in $(r, p, \mathbf{T}, S)$ above.

$(-, \mathbf{p}, \mathbf{t}, -)$: →② with $\mathcal{A} = \{A\}$, i.e., a matching of A
Each event is a vertex in A, each eligible teacher-period combination is represented as a

vertex in B. By setting $\mathcal{A} = \{A\}$ we ensure that each combination can be assigned at most once. Unary constraints are modeled by (non-)existence of edges.

$(-, \mathbf{p}, T, -)$, $(-, \mathbf{p}, -, S)$: $\textcircled{3}\!\rightarrow$
If $(G = (A \cup B, A \times B), \mathcal{A})$ is an instance of $\textcircled{3}$ we regard A as events, B as periods and edges as (full) availabilities of events. Each $A_i \in \mathcal{A}$ represents a teacher or student, respectively. Teachers are available for all periods as well. Observe that these subproblems are already NP-complete if each event has only two teachers that are always available, or two students.

$(r, \mathbf{p}, t, -)$: $\textcircled{6}\!\rightarrow$
If $(G = (A \cup B, E), \mathcal{A}, \mathcal{A}')$ is an instance of $\textcircled{6}$ we regard A as events, B as periods and edges as availabilities of events. Moreover, we assign room i to each event $a \in A_i$ and teacher j to all events $a \in A'_j$, both with full availability.

$\square$

With both Theorems 3.1 and 5.1 we now have the hard/easy-classification of all subproblems as depicted in Tables 3 and 4.

Table 3: Efficiently decidable subproblems.

Subproblems $\mathcal{P}$	$\preceq \mathcal{P}'$	Reduction	No. of subproblems
$(*, *, \mathbf{T}, *)$, $(*, *, \mathbf{t}, *)$	$(r, p, \mathbf{T}, S)$	$\rightarrow\textcircled{2}$	16
$(*, \mathbf{p}, -, -)$	$(r, \mathbf{p}, -, -)$	$\rightarrow\textcircled{2}$	2
$(-\mathbf{p}, t, -)$	$(-\mathbf{p}, \mathbf{t}, -)$	$\rightarrow\textcircled{2}$	2
$(\mathbf{r}, \mathbf{p}, -, -)$	$(\mathbf{r}, \mathbf{p}, t, -)$	$\rightarrow\textcircled{11}$	2
$(\mathbf{r}, *, *, *)$,$(\mathbf{r}, *, \mathbf{T}, *)$, $(\mathbf{r}, *, \mathbf{t}, *)$	$(\mathbf{r}, p, \mathbf{T}, S)$	$\rightarrow\textcircled{2}$	20
			42

Table 4: NP-complete subproblems.

Reduction	Subproblem $\mathcal{P}$	$\preceq \mathcal{P}'$	No. of subproblems
$\textcircled{3}\!\rightarrow$	$(-, \mathbf{p}, T, -)$	$(*, \mathbf{p}, T, *)$, $(*, \mathbf{p}, \mathbf{T}, *)$, $(\mathbf{r}, \mathbf{p}, T, *)$	10
$\textcircled{3}\!\rightarrow$	$(-, \mathbf{p}, -, S)$	$(*, \mathbf{p}, *, S)$	6
$\textcircled{6}\!\rightarrow$	$(r, \mathbf{p}, t, -)$	$(r, \mathbf{p}, \mathbf{t}, *)$, $(\mathbf{r}, \mathbf{p}, \mathbf{t}, *)$, $(\mathbf{r}, \mathbf{p}, *, S)$	8
			24

For practical implementations the reductions in Table 3 provide polynomial-time algorithms indeed, however some of them are just as easy as the very basic assignment problem BAP: $(\mathbf{r}, -, -, -)$, $(-\mathbf{p}, -, -)$, $(-, -, \mathbf{t}, -)$ and $(\mathbf{r}, -, \mathbf{t}, *)$ can already be captured by problem $\textcircled{1}$. Similarly, we can provide bipartite assignment problems as natural upper bounds that capture the NP-complete subproblems, which we show without further proof in Table 5. So an algorithm for a single problem $\textcircled{i}$ in this table can be used to solve the UCT subproblems in the same row in a straightforward way.

Table 5: NP-complete subproblems and how they can be modeled.

subproblems	Reduction
$(-, \mathbf{p}, T, -)$, $(-, \mathbf{p}, -, S)$, $(\mathbf{r}, \mathbf{p}, T, -)$, $(r, \mathbf{p}, -, S)$, $(-, \mathbf{p}, \mathbf{t}, S)$	$\rightarrow$④
$(r, \mathbf{p}, *, *)$	$\rightarrow$⑦
$(\mathbf{r}, \mathbf{p}, \mathbf{t}, *)$	$\rightarrow$⑩
$(r, \mathbf{p}, \mathbf{t}, -)$	$\rightarrow$⑫
$(r, \mathbf{p}, \mathbf{t}, S)$	$\rightarrow$⑬

6 Additional Constraints On Teachers and Rooms

In this section we look at some variants of UCT subproblems which occur by adding constraints w.r.t. teachers and rooms. We consider the following three constraints, which sound are quite similar but have different effects on complexities:

C_{10} *An event can only have a room that is eligible for the assigned teacher.*
Each teacher has a set of rooms where she can teach, regardless of which event. For example, some teachers require rooms which are easily accessible.

C_{11} *An event can only have a room that is eligible for this event.*
Each event has a set of suitable rooms we can choose from. For example, some events require a room with certain technical equipment.

C_{12} *An event can only have a room that meets the room restictions of the assigned teacher for this event.*
Each teacher defines a set of rooms per event where she can teach, e.g., some teachers need a blackboard for certain events while others require two projectors for the same event.

Constraints C_{10} and C_{12} need to be considered if both rooms and teachers are part of the subproblem notation, while C_{11} is active in all room-assignment problems. In Table 6 we list all relevant subproblems that were in P in the hitherto model and where we now add one of the above constraints. Additionally we mention the respective bipartite assignment problems used in the reductions. The complexity status of subproblem $(\mathbf{r}, p, \mathbf{T}, -)$ remains open in one case.

Table 6: Subproblems and their complexity if either constraint C_{10}, C_{11} or C_{12} is added.

Subproblems	$+\ C_{10}$	$+\ C_{11}$	$+\ C_{12}$
$(\mathbf{r}, p, T, *)$	P $\rightarrow$②	P $\rightarrow$②	P $\rightarrow$②
$(r, p, \mathbf{T}, *)$	P $\rightarrow$②	P $\rightarrow$②	P $\rightarrow$②
$(\mathbf{r}, -, \mathbf{t}, *)$	P $\rightarrow$②	P $\rightarrow$②	P $\rightarrow$②
$(\mathbf{r}, -, \mathbf{T}, *)$	P	P $\rightarrow$②	P
$(\mathbf{r}, p, \mathbf{t}, -)$	P $\rightarrow$⑪	P $\rightarrow$②	NPC ⑨$\rightarrow$
$(\mathbf{r}, p, \mathbf{T}, -)$	?	P $\rightarrow$②	NPC ⑨$\rightarrow$
$(\mathbf{r}, \mathbf{p}, t, -)$	P $\rightarrow$⑪	NPC ⑫$\rightarrow$	NPC ⑫$\rightarrow$

We mention indications of some reductions to justify complexities.

C_{10}, C_{11}, C_{12}: $(\mathbf{r}, p, T, *)$ $\rightarrow$②:
Edges in ② exists if the room is eligible with regard to sufficient capacity, the assigned teacher (C_{10}), the event (C_{11}) or the teacher-event combination (C_{12}).

C_{10}, C_{11}, C_{12}: $(r, p, \mathbf{T}, *) \to \textcircled{2}$:
Edges in $\textcircled{2}$ exists if the teacher has ability to teach an event and the room is eligible for him (C_{10}), for the event (C_{11}) or for the given teacher-event combination (C_{12}).

C_{10}, C_{11}, C_{12}: $(\mathbf{r}, -, \mathbf{t}, *) \to \textcircled{2}$:
If C_{10} or C_{12} are activated vertices in B represent an eligible room-teacher combination. Edges are present if a teacher has ability to teach an event and the room has sufficient capacity C_{10}. For activated C_{12} we must additionally ensure that the teacher-room combination is eligible for the respective event. If C_{11} is activated rooms and teachers do not share constraints and hence they can be calculated independently. Observe non existence of edges in case of the room assignment if an room is not eligible for an event.

C_{10}, C_{12}: $(\mathbf{r}, -, \mathbf{T}, *)$:
We sketch algorithm which decides this problem in $O(\#E \cdot \#R \cdot \#T)$. For each event, we try a room and check, if this room is eligible for at least $\#T_e$ many teachers which have ability to teach this event. If so, we move on with the next event. If not, we try the next room and ask again. If this is not successful for an event, we return a 'no'-answer.

C_{10}: $(\mathbf{r}, p, \mathbf{t}, -) \to \textcircled{11}$:
For each period we construct an instance of BMP($\mathcal{A}$;$\mathcal{B}$) which is closed under product as follows: We set $\mathcal{A} = \{A\}$ since all events in a common period are pairwise conflicting. Eligible teacher-room combinations are partitioned in $\mathcal{B}$ according to common teachers. Eligible in this case means that a teacher and room are available in the considered period and the room is in the teacher's room set. Edges are drawn between some event and a teacher-room combination if the respective teacher is able to teach the event. Note that if a teacher has ability to teach an event then the event is connected to all rooms of that teacher, which shows closure under product.

C_{10}: $(\mathbf{r}, \mathbf{p}, t, -) \to \textcircled{11}$:
We partition all events in $\mathcal{A}$ according to common teachers. All eligible room-period combinations are partitioned in $\mathcal{B}$ according to common periods. Edges between events and room-periods exist if an event and its assigned teacher are available in this period and the room is in the teacher's room set. Note that if events of the same teacher are available in a same period then all those events are connected to the same room-period combinations in this period block, so closure under product holds.

C_{11}/C_{12} :$\textcircled{12} \to$ $(\mathbf{r}, \mathbf{p}, t, -)$
We regard $\mathcal{A}$ as events partitioned to common teachers and $\mathcal{B}$ as room-periods combinations partitioned to common periods. Edges between an event and a room-period combination are regarded as:

- the event is available in this period,
- the teacher is available in this period,
- if C_{11} is activated: the room is eligible for this event
- if C_{12} is activated: the room is eligible for the teacher in combination with this event

Observe that in this case we can not obtain closure under product.

C_{12}: $\textcircled{9} \to$ $(\mathbf{r}, p, \mathbf{t}, -)$:
If $(G = (A \cup B, E), \{A\}, \mathcal{B}, \mathcal{B}')$ is an instance of $\textcircled{9}$ with partitions $\mathcal{B}$, $\mathcal{B}'$ we regard A as events scheduled in a common period. Each vertex in B is regarded as a room-teacher

combination and edges indicate that a teacher-room combination is eligible for an event. That is, if the teacher has ability to teach this event, she is available and the respective room is in her set of eligible rooms for this event. All room-teacher combinations are partitioned w.r.t. common teachers and common rooms, respectively. Since all events are scheduled in parallel, each teacher and each room can be assigned at most once.

7 Conclusion

We have settled the complexities of all subproblems of a UCT model with a typical set of constraints. Our decomposition approach is such that subproblems can be understood as re-computation problems, as building blocks for algorithms or as consistency checks for the general task. To obtain this classification we have carried out the following program of investigations:

(1) Identify an order relation on subproblems that is consistent with polynomial-time many-one reductions (Sec. 3).

(2) Use abstract models (Sec. 4) to settle the complexity of all subproblems with help of maximal and minimal subproblems w.r.t. this relation (Sec. 5).

(3) Capture groups of subproblems with the same model and identify common computational tasks (Sec. 5).

(4) Analyse the effect of variations of constraints on selected subproblems (Sec. 6).

We think that apart from our concrete results which may appear in several UCT scenarios, in particular (4) gives a strong motivation to look for algorithms for some of the specialized bipartite assignment problems from Section 4. Future research could also comprise to work on some of the limitations of our initial model. So we can ask how student sectioning can fit into such an approach, and how other constraints can be included. For example, limiting the workload of teachers can be modeled by limiting the number of edges from certain sets of edges in an assignment (cf. [TIR78]).

Another line of research can be to determine the borderline between easy and hard problems more precisely in terms of parameters that are relevant to practical applications: Note for example that problem $\mathrm{BAP}(\mathcal{A})$ with an arbitrary number α of conflicting sets is hard, but we know that it is easy if $\alpha \leq 3$. Since α corresponds in some scenarios to the number of different curricula one can ask if $\mathrm{BAP}(\mathcal{A})$ is still hard for any fixed α. Another example are multiple-assignment subproblems where the assignment of a single teacher is easy, while the same task is hard if events can have multiple (and yet unbounded number of) teachers. Finally, we also believe that our approach (1)-(4) can serve as a blueprint to analyse the structure of UCT models with other sets of constraints in order to get more insight into different what-if scenarios.

References

[AdW02] A.S. Asratian and D. de Werra. A generalized classteacher model for some timetabling problems. *European Journal of Operational Research*, 143(3):531 – 542, 2002.

[BKH15] Hamed Babaei, Jaber Karimpour, and Amin Hadidi. A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, 86:43–59, 2015.

[CK96] Tim B. Cooper and Jeffrey H. Kingston. The complexity of timetable construction problems. In Edmund Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling*, pages 281–295, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[COS01] Richard Cole, Kirstin Ost, and Stefan Schirra. Edge-Coloring Bipartite Multigraphs in O(E logD) Time. *Combinatorica*, 21(1):5–12, Jan 2001.

[Csi65] J. Csima. *Investigations on a Time-table Problem.* Ph.D. thesis, School of Graduate Studies, University of Toronto, 1965.

[DPS16] M. Dostert, A. Politz, and H. Schmitz. A complexity analysis and an algorithmic approach to student sectioning in existing timetables. *Journal of Scheduling*, 19(3):285–293, 2016.

[dW71] D. de Werra. Construction of school timetables by flow methods. *INFOR Journal*, 9(1):12–22, 1971.

[dW03] Dominique de Werra. Constraints of Availability in Timetabling and Scheduling. In Edmund Burke and Patrick De Causmaecker, editors, *Practice and Theory of Automated Timetabling IV*, pages 3–23, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[EIS75] S. Even, A. Itai, and A. Shamir. On the Complexity of Time Table and Multicommodity Flow Problems. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, SFCS '75, pages 184–193, Washington, DC, USA, 1975. IEEE Computer Society.

[FF87] L. R. Ford and D. R. Fulkerson. *Maximal Flow Through a Network*, pages 243–248. Birkhäuser Boston, Boston, MA, 1987.

[GJ90] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1990.

[Got62] C. C. Gotlieb. The Construction of Class-Teacher Time-Tables. In *IFIP Congress*, pages 73–77, 1962.

[HK73] John E. Hopcroft and Richard M. Karp. An n5/2 Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[Hol81] Ian Holyer. The NP-Completeness of Edge-Coloring. *SIAM J. Comput.*, 10(4):718–720, 11 1981.

[Kar72] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[Kin14] Jeffrey H. Kingston. Timetable construction: the algorithms and complexity perspective. *Annals of Operations Research*, 218(1):249–259, Jul 2014.

[Kos05] Philipp Kostuch. The University Course Timetabling Problem with a Three-Phase Approach. In Edmund Burke and Michael Trick, editors, *Practice and Theory of Automated Timetabling V*, pages 109–125, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[KS13] Simon Kristiansen and Thomas Riis Stidsen. A Comprehensive Study of Educational Timetabling - a Survey. Report 8.2013, Department of Management Engineering, Technical University of Denmark, 2013.

[LL12] Gerald Lach and Marco E. Lübbecke. Curriculum based course timetabling: new solutions toUdine benchmark instances. *Annals of Operations Research*, 194(1):255–272, Apr 2012.

[Mar05] Dniel Marx. NP-completeness of list coloring and precoloring extension on the edges of planar graphs. *Journal of Graph Theory*, 49(4):313–324, 2005.

[McC06] Barry McCollum. University Timetabling: Bridging the Gap between Research and Practice. In *in Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*, pages 15–35. Springer, 2006.

[PZ80] David A. Plaisted and Samuel Zaks. An NP-complete matching problem. *Discrete Applied Mathematics*, 2(1):65–72, 1980.

[Rud15] Hanna Rudova. University Course Timetabling - From Theory to Practice. In *Multidisciplinary International Scheduling Conference (MISTA 2015) (Talk)*, Prague, Czech Republic, 2015.

[Sch16] David Schindl. Student sectioning for minimizing potential conflicts on multi-section courses. *Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling (PATAT 2016)*, pages 327–337, 2016.

[tEW01] H. M. M. ten Eikelder and R. J. Willemen. Some Complexity Aspects of Secondary School Timetabling Problems. In Edmund Burke and Wilhelm Erben, editors, *Practice and Theory of Automated Timetabling III*, pages 18–27, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[TIR78] Steven L. Tanimoto, Alon Itai, and Michael Rodeh. Some Matching Problems for Bipartite Graphs. *J. ACM*, 25(4):517–525, October 1978.