A Rule Language and Feature Model for Educational Timetabling[*]

Corentin Behuet, Vincent Barichard, David Genest, Marc Legeay, and David Lesaint

Univ Angers, LERIA, SFR MATHSTIC, F-49000 Angers, France
{firstname.lastname}@univ-angers.fr

Abstract. Educational timetabling subsumes core problems (student sectioning, course scheduling, etc.) which are challenging from a modeling and computational perspective. In this paper, we expand on the University Timetabling Problem framework (UTP) designed to address a wide range of university timetabling problems. The framework combines a rich data schema with a rule language and comes with a tool chain to compile instances into constraint satisfaction problems. We present the UTP modeling language and a feature model to capture the problem classes that are expressible. The feature model provides a simple problem classifier which we use in our literature review. We also present a timetabling instance generator and report on experiments carried out with Constraint Programming, Answer-Set Programming and Mixed Integer Linear Programming solvers.

Keywords: Timetabling, Domain-Specific Modeling Language, Feature Model, Exact Methods, Timetable Dataset Generation

1 Introduction

Various problem formulations, data formats and algorithms have been proposed to tackle specific aspects of university timetabling ranging from curriculum balancing [16,18,50], student sectioning [42,52], examination timetabling [14,10,40], curriculum-based or post-enrollment-based course timetabling [40,12,37,13,26,17], tutor allocation [15], to minimal timetabling perturbation [38,36]. Modeling languages have also been developed, notably the XHSTT language [48], the ITC language used in the 2019 international timetabling competition [41,29] and the UTP language introduced in [9].

UTP is a modeling language for educational timetabling problems which is built on a structured domain model coupled with a rules language. The model supports sessions requiring a single resource and those needing multiple resources capturing essential limitations related to the timing of sessions and distribution of resources. It operates under the presumption that resources can overlap (i.e. rooms, teachers, and students can be involved in simultaneous sessions), though this approach can be adjusted through specific scheduling rules that prevent such overlaps. Given a UTP instance, the objective is to assign time slots and allocate resources to class sessions so that core constraints and rules are satisfied.

We first introduce the UTP schema which has been extended to broaden the range of problems that can be modeled. We then present a feature model to classify educational

[*]Supported by a research grant from Université d'Angers.

timetabling problems and compare UTP with other modeling languages. Lastly, we report on experiments carried out with 3 UTP solvers - CP, ASP, MIP - on instances created with a custom generator.

2 The UTP Schema

The UTP schema combines a schema to model timetabling entities and solutions, and a rule language. The entity schema models the entities of a UTP instance - scheduling horizon, resources, and course elements including course sessions -, their properties and relationships. The rule language is user-oriented and serves to concisely express constraints over any set of entities on the different facets of a problem (e.g., session scheduling, capacity planning, resource allocation). Rules are formulated using a catalog of timetabling constraint predicates and a query language to select, filter and bind entities to sessions.

A rule-based UTP instance is converted to a constraint-based instance that is readily processable by solvers. The conversion translates the entity schema as decision variables and core constraints, and then flattens rules as additional constraints. A UTP instance is thus cast as a hard constraint satisfaction problems. Solving an instance involves scheduling sessions and assigning them resources so that the core constraints and the rule constraints are satisfied. The solution schema allows to represent any timetabling solution computed for an instance, be it incomplete or inconsistent.

This section introduces the components of the schema. The abstract syntax of the entity schema is given in Table 1, its constraint-based modeling in Table 6 and Table 7 (Appendix), and the syntax of the rule language and constraint predicates in Table 8, Table 9 and Table 10 (Appendix).

2.1 Entity Schema

The entity schema of a UTP instance combines a hierarchy of course elements (i.e., courses, course parts, part classes and class sessions) a scheduling horizon over which sessions are to be scheduled, and 4 types of resources to which sessions must be allocated to (i.e., rooms, teachers, students and student groups). The schema encodes the nesting of course elements and various properties and constraints concerning session scheduling, resource availability, resource eligibility, teaching service, room capacity, and student sectioning.

The scheduling horizon is a range of integers denoting time points. The time points are the start and end times allowed for sessions and any duration (i.e., session length, travel time and break time) is measured as a number of time points. The horizon is defined using 3 instance fields: the number of weeks w dividing the horizon, the number of weekdays d making a week and the number of daily slots m making a 24-hour day. The time points correspond to all possible triplets combining a week, a weekday, and a daily slot. Note that daily slots may have any granularity (e.g., 1 minute, 2 hours) and the scheduling horizon may be sparse (e.g., if weeks i and $i+1$ are not consecutive calendar weeks for some $1 \leq i < w$ or if weekdays are dropped, i.e., $d < 7$).

Course elements follow a hierarchical structure. Each course (e.g., Algorithms) consists of one or more parts (e.g., Lecture and Lab), each part is taught to one or more classes (e.g., lecture classes A and B), and all classes of a part have the same number of sessions (e.g., sessions 1 to 10 for each lecture class). The schema requires that all sessions in a class have the same duration and be chronologically ranked, i.e., session of rank $i + 1$ in a class must start after session of rank $i - 1$ ends in any solution. These constraints are paramount to model course plans that rely on clear-cut sessions (e.g., starting lab classes after 2 lecture sessions, synchronizing the 5^{th} sessions of lab classes for a joint examination). Besides, the schema allows to restrict the possible time slots for the sessions of a part by setting allowed and forbidden ranges using the time format. Note that sessions must start and end on the same day, and cannot be interrupted. The schema also specifies a set of possible resources for each session. As for students, a sectioning plan is assumed and hard-coded together with group-to-class assignments. Specifically, students are partitioned into groups, and groups aggregated and assigned to classes with no group being assigned to more than one class per course part. The schema encodes group and class headcounts as well as class headcount thresholds used for sectioning. Other sectioning data and constraints are compiled away. The implicit constraint to satisfy is that a group must attend all the sessions of a class it is bound to.

Teacher-to-session assignments are not fixed but subject to domain and cardinality constraints. To meet practical needs, the schema allows multiple teachers per session (e.g., joint supervision of a lab session) and teacher-less sessions (e.g., unsupervised project work). The number of teachers per session is specific to each course part and is lower- and upper-bounded, possibly fixed. Each part is also associated with a set of required teachers and a superset of allowed teachers. Hence two sessions of a class may be allocated different teachers and numbers of teachers. A part also sets the fixed number of sessions a teacher is committed to. Overall, various demand and capacity requirements relating to teaching service can be addressed on course parts. If needed, finer-grained rules may be imposed (e.g., requiring the same staff for a class, naming a lecturer for a session).

Similarly, each part sets the required and possible rooms for its sessions and their number. This caters for the case of multi-room sessions (e.g., for hybrid teaching) and room-less sessions (e.g., field trips). In addition, each part casts its sessions as room-exclusive or room-inclusive which entails different allocation constraints. A session is room-exclusive if none of the room(s) hosting it may simultaneously host another session. Conversely, a room-inclusive session allows for its room(s) to be shared from start to finish. While single-room sessions may be cast as exclusive or inclusive, multi-room sessions may only be cast as exclusive. That is, every session of a part whose room upper-bound is greater than 1 is considered exclusive. The rationale is that multi-room inclusive sessions have arguably little practical interest and they also burden the computational model with decisions to make on the distribution of groups in shared rooms.

All resources enforce capacity constraints w.r.t. their utilization. Students, teachers and rooms are considered cumulative resources in this respect. That is, they may attend, teach or host simultaneous sessions. A cumulative model is paramount to satisfy flexible attendance requirements (e.g., students attending tutoring sessions overlapping with compulsory courses) and multi-class events (e.g., an amphitheater hosting a joint conference for different classes). Again, rules may be used to impose disjunctive resources or to ban session overlapping. No limit is set on the number of parallel sessions teachers and students may attend. Session hosting however is subject to capacity constraints and the schema encodes the capacity of each room, allowing for infinite capacity to handle virtual rooms. As discussed above, at any point in time, an allocated room will either co-host a multi-room (exclusive) session or host one or more single-room sessions (one only if a session is exclusive). The schema hence enforces two kinds of capacity constraints. The single-room case involves checking if the total headcount of the session(s) falls below the room capacity. The multi-room case involves ensuring the total capacity of the rooms envisaged for the session exceeds its headcount. If so, no restriction is imposed as to the distribution of students in rooms and whether it preserves group structure or not.

Lastly, the schema provides users with the ability to define their own classes of entities, mixing course elements and resources as needed with no limit on classification (e.g., a block of rooms, the lecturers of a faculty department). This is achieved by labeling entities. Labels, built-in entity types and ids are the building blocks of the query language to forge rules for any group of entities.

Table 1 provides a formal specification of the schema elements. Resources and course elements, except sessions, are referred to as *entities*. Entities are typed, the set of sessions is cast as distinct type, and each type is modeled as a finite set. The course element hierarchy defines 1-to-many *composition relations* over the pair of types (X, Y) corresponding to parent and child types in the course element hierarchy. Each relation is modeled by a function $d^{X,Y} : X \to 2^Y$ mapping each object i of type X to the set $d_i^{X,Y}$ of its constitutive objects of type Y. For instance, $d^{P,K}$ models the classes of each part. Each *compatibility relation* defining the allowed or assigned resources of a course element object for a given resource type and course element type defines a many-to-many relation which we model the same way. For instance, $d^{P,R}$ models the allowed rooms per part and $d^{K,G}$ the set of groups assigned to classes.

For notational convenience, the table also defines the maps resulting from the symmetric and transitive closure of the binary relation merging the composition and compatibility maps. This includes the maps computed over the course tree. For instance, $d^{K,P}$ models the (singleton) part of each class, and $d^{C,S}$ the sessions of a course. This also includes the inverse compatibility constraints and those inherited along the course tree. For instance, $d^{S,R}$ models the rooms allowed for a session which results from the composition of $d^{S,K}$, $d^{K,P}$ and $d^{P,R}$. Lastly, the table defines the constants (e.g., number of weeks), scalar properties (e.g., room capacity), and remaining relations and sets (e.g. required resources, labels).

2.2 Solutions

The solution schema is used to encode any solution pre-computed for an instance. Such a solution needs not be complete, nor consistent with the instance constraints. An instance may hence be associated with any kind of input solution based on the computational task, e.g. no solution at all when generating a timetable from scratch, a resource allocation solution to extend into a complete timetable, a seed solution to

w	number of weeks dividing the scheduling horizon
d	number of weekdays making a week
m	number of daily slots making a 24-hour day
$W = \{1, \ldots w\}$	range of weeks
$D = \{1, \ldots d\}$	range of weekdays
$M = \{1, \ldots m\}$	range of daily slots
$H = \{1, \ldots w \times d \times m\}$	range of time points (schedule horizon)
C	courses
P	course parts
K	part classes
R	rooms
T	teachers
U	students
G	groups of students
$\Gamma = \{C\}$	course domain
$\mathcal{E} = \{\Gamma, C, P, K, R, T, U, G\}$	types of entities
$E = \cup_{X \in \mathcal{E}} X$	set of entities
$d_i^{X,Y} \subseteq Y, X \in \{\Gamma, C, P, K\}$	set of entities of type Y tied to entity i of type X
$d_i^{X,Y} \subseteq Y, X \in \{R, T, U, G\}$	set of entities of type Y associated with entity i of type X
$\mathcal{L} \subseteq 2^E$	labels
$S^{(e)}$	exclusive class sessions
$S^{(i)}$	inclusive class sessions
$S = S^{(e)} \uplus S^{(i)}$	class sessions
$d_s^{S,H} \subseteq H$	start times allowed for session s
$d_s^{S,X} \subseteq X$	set of entities of type X tied to session s
$d_i^{X,S} \subseteq S$	set of sessions tied to entity i of type X
$\underline{d}_i^{X,S} \subseteq S$	set of sessions required by resource entity i of type X
$min\_rooms_p^P \in \mathbb{N}$	min number of rooms usable by each session of part p
$max\_rooms_p^P \in \mathbb{N}$	max number of rooms usable by each session of part p
$min\_lecturer_p^P \in \mathbb{N}$	min number of lecturers usable by each session of part p
$max\_lecturer_p^P \in \mathbb{N}$	max number of lecturers usable by each session of part p
$size_g^G \in \mathbb{N}$	headcount of group g
$size_k^K \in \mathbb{N}$	headcount of class k
$capacity_r^R \in \mathbb{N}$	capacity of room r
$length_s^S \in H$	duration of session s
$rank_s^S \in \mathbb{N}^*$	rank of session s in its class
$service_{t,p}^{T \times P} \in \mathbb{N}$	number of sessions required by teacher t in part p

Table 1: Core data model.

improve, or an inconsistent solution to repair. Formally, the solution schema supports the representation of any decision made for a session as to its start time, its set of rooms and its set of teachers.

2.3 Predicates, Constraints and Rules

The UTP schema comes with a rule language to formulate instance-specific constraints. Rule constraints add to the built-in constraints of the schema and all must be checked when evaluating a solution. The rule language is designed to target groups of entities, or individual entities, and constrain the scheduling of their sessions from any standpoint (e.g., an institutional rule imposing a time structure on curricula, a disjunctive scheduling rule applied to student groups, a rule modeling the service plan within a faculty department, a rule for a lecturer's agenda). The schema comes with a catalog of timetabling predicates to build rules and compile them into constraints. It also includes a query language to select entities and sessions on which rules should apply.

All these components are designed around the concept of *e-map*. Formally, an e-map is a pair (e_i, S_i) mapping an entity e_i to a set S_i of sessions. The query language is used to forge queries that retrieve sets of e-maps. Each query selects, filters and binds entities to sessions from instance data in order to extract one or more sets of e-maps. Each rule is bound to a predicate and scoped by a query. At flattening time, the query is performed to retrieve a fixed number of sets of e-maps. The rule is then compiled into a conjunction of constraints by computing the cross-product of the extracted sets and applying the predicate to each tuple of e-maps in the cross-product. Constraint e-maps act as guards when checking solutions and they also narrow the scope of interpretation. The rationale is to discard constraints that are irrelevant (e.g., a teacher's constraint forbidding afternoon lectures while the solution only assigns him lab sessions) and, more generally, to limit constraint checks to the proposed assignments (e.g., checking the above lecturer's constraint on the actual lectures the solution assigns him).

As mentioned above, each constraint applies a predicate to a tuple of e-maps. UTP predicates either accept a fixed number of e-maps or are variadic. Their semantics may be indifferent to the ordering of their arguments or not, and some accept parameters. Besides, each predicate may be used indistinctly with course e-maps or resource e-maps (i.e., e-maps pairing course elements or resources), and any n-ary constraint may freely mix the two types (e.g., a constraint booking rooms for sessions involving different classes). Let $F = E \times 2^S$ denote the domain of e-maps, the general form of a constraint is $c((e_1, S_1), \ldots, (e_n, S_n), p_1, \ldots, p_m)$ where c is a predicate of arity n, $(e_1, S_1), \ldots, (e_n, S_n)$ are e-maps ($(e_i, S_i) \in F$ for $i = 1 \ldots n$) and $p_1, \ldots, p_m$ are values for the parameters of c ($m \geq 0$).

The semantics of constraints relies on a join operation between constraint e-maps and solutions. Note first that any solution may be cast as a tuple of e-maps by converting the session-to-resource assignments into resource e-maps and re-encoding the fixed maps binding course elements to their sessions. We say an e-map is *null* if it pairs an entity with an empty set of sessions, and, by extension, a tuple of e-maps is null if it includes a null e-map. Given a solution and an e-map for some entity, we call *joint e-map* the pairing of the entity with the set of sessions on which the solution and the e-map agree, i.e., the set-intersection of the sessions of the e-map and those assigned/bound to the entity in the solution encoding. We say a solution is *inconsistent* with an e-map if their joint e-map is null. The join operation extends to tuples by performing the operation component-wise and a solution is said to be inconsistent with a tuple of e-maps if its is inconsistent with at least one e-map in the tuple.

The evaluation of a solution against a constraint is conditioned by the tuple of e-maps joining those of the solution and the constraint. If the joint tuple is null, the constraint is considered satisfied (i.e., it is deemed irrelevant and discarded). Otherwise, the predicate is evaluated on the joint e-map and the result depends on its built-in semantics. Specifically, the predicate is assessed on the tuple of sets obtained by substituting each set of sessions in the joint e-map either by the set of their assigned start times, or the set of their assigned resources of a given type (rooms, etc). Which type (time or resource type) to pick per e-map is fixed and predicate-specific (e.g., a temporal predicate will substitute any e-map argument by start times). Note that entities play no role in the evaluation once the join and substitution operations are over: each predicate is ultimately evaluated on sets made of start times or sets of resources. Note also that join operations leave course e-maps unchanged unlike resource e-maps. This means constraints applying exclusively to course e-maps are de facto unconditional.

The UTP catalog provides predicates to cover the various dimensions of time–tabling problems. Some only address scheduling (i.e., start times), others room allocation, and so on. Table 9 and Table 10 given in Appendix describe the predicates of the catalog and provide their semantics. Syntactically, each rule binds a predicate to a query and denotes the conjunction of constraints obtained by applying the predicate to each tuple of e-maps extracted by the query. A rule has the form $c\langle Q, p_1, \dots, p_m\rangle$ and is interpreted by the formula

$$\forall (e_1, S_1) \in [\![Q_1]\!], \dots, (e_n, S_n) \in [\![Q_n]\!] : c((e_1, S_1), \dots, (e_n, S_n), p_1, \dots, p_m)$$

where c is a predicate of arity n accepting m parameters ($m \geq 0$), Q is a query sized to extract n sets of e-maps, $[\![Q_i]\!]$ denotes the i-th set of e-maps extracted with Q ($i = 1 \dots n$), and $p_1, \dots p_m$ are values for the parameters of c.

3 A Feature Model

This section introduces a feature model for educational timetabling problems based on the UTP schema. The model is not meant to be exhaustive, nor stable, but is a first attempt to capture the key variability points (the features) in the family of instances that can be expressed with the schema. Some features are plain flags characterizing the compliance of an instance to the schema (e.g., whether courses are hierarchically structured or not) while others are logical assertions on instance data (e.g., whether the number of weeks is set to 1 or not). In either case, each feature may be checked against any instance and, in turn, instances classified into different classes based on the features they satisfy.

The feature model hence decomposes the space of UTP problems which serves different purposes. One is to quickly assess whether the schema is applicable to a particular setting. Another is to provide a straightforward characterization of problem classes, similarly to the way 3-field notation is used in other scheduling domains [27,3,1].The aim is also to facilitate the comparison of UTP with competing schemas, possibly paving the way for formal reductions between problems and conversions between schemas. Lastly, the feature model can guide the configuration of efficient computational models by using features to reformulate or optimize built-in constraints and predicate implementations.

We first recall the basic notations and definitions commonly used in feature modeling languages [31,19,44]. A feature model is a tree-like structure connecting features and factoring in different feature configurations. A configuration is a subset of features selected from the model. The configuration process is subject to constraints that primarily capture dependencies that exist between a feature and its children (a.k.a., sub-features). These fall into 4 categories: *mandatory sub-feature* (it must be selected if the parent is) labeled by •, *optional sub-feature* (it may be selected if the parent is) labeled by ∘, *or-feature* (at least one of the sub-features must be selected) labeled by +, and *xor-feature* (exactly one sub-feature must be selected) labeled by 1. Finer-grained cardinality constraints may apply as well as cross-tree constraints modeling dependencies or incompatibilities between features that sit in different branches.

Table 2 details our feature model. The feature-tree (rotated anticlockwise by 90°) has 3 levels: the root node (not shown), its sub-features and their labels shown respectively on the 2nd and 1st columns and their variants shown on the next 2 columns. For instance, selecting feature hosting in a configuration requires selecting at least one of no-room, single-room or multi-room. The last column provides the formal or informal characterization of each leaf feature. The sub-features of the root characterize core structural elements (course and time structure), orthogonal decision layers (scheduling, room allocation, etc.), and cross-cutting concerns (session planning, resource availability, etc.). The latter is tagged optional and so are hosting and teaching as these decision layers may be out of scope in an instance. We explain next the variants of these sub-features.

course-hierarchy applies to instances whose course elements are nested hierarchically. event applies when events unrelated to courses (e.g., staff meetings) must be scheduled too. The next 3 features characterize the sparsity and scope of the time horizon. full-period indicates if it is built on consecutive calendar weeks and full-week if a weekday is missing. single-week checks whether the instance is restricted to a single week which is typical of timetabling practices in high schools. The next 3 characterize the temporal structure imposed on sessions from "time grids" in high-schools to free-flow timetables for higher grade curricula. no-overlap holds true if sessions can never overlap if they start at different times, same-duration if all sessions have the same duration, and modular if every session length, break time included, breaks down to a unit session length (e.g., some sessions are 1h long and any other session is measured in hours).

The next features characterize room utilization. no-room, single-room, multi-room, hold true if the instance includes a session that demands no room, a single room or more than 1 room, respectively. Similar features are introduced for the demand on teachers. all-exclusive, none-exclusive, some-exclusive, indicate if the instance includes only room-exclusive sessions, only inclusive sessions or a mix, respectively. room-capacity, service, and sectioning apply if resp. room capacity, teaching service and student sectioning are in scope. As for teaching, session-overlap indicates if teachers are cast as disjunctive resources (the counterpart is introduced for students). Lastly, the sub-features of crosscutting capture cross-cutting concerns and we simply list examples of constraints taken from the UTP catalog to convey the meaning.

●	courses	○	course-hierarchy	*courses are decomposed hierarchically into sessions*
			event	*events unrelated to courses must be scheduled*
●	timing	○	full-period	*weeks are consecutive calendar weeks*
			full-week	$d = 7$
			single-week	$w = 1$
●	scheduling	○	no-overlap	$\forall s_i, s_j \in S, s_i \neq s_j, \forall h_i \in d_{s_i}^{S,H}, \forall h_j \in d_{s_j}^{S,H}$ $h_i < h_j \wedge h_i \div m = h_j \div m :\ h_i + length_{s_i}^{S} \leq h_j$
			same-duration	$\forall s_i, s_i \in S, length_{s_i}^{S} = length_{s_i}^{S}$
			modular	Let $A = \{h_j - h_i \mid s_i, s_j \in S, h_i \in d_{s_i}^{S,H}, h_j \in d_{s_j}^{S,H},$ $s_i \neq s_j,\ h_i < h_j\} : gcd(A) = \min(A) \wedge gcd(A) > 1$
○	hosting	+	no-room	$\exists p \in P,\ min\_rooms_p^P = max\_rooms_p^P = 0$
			single-room	$\exists p \in P,\ min\_rooms_p^P = max\_rooms_p^P = 1$
			multi-room	$\exists p \in P,\ min\_rooms_p^P \geq 1 \wedge max\_rooms_p^P > 1$
		○	room-capacity	$\forall r \in R,\ capacity_r^R \neq \emptyset$
		1	all-exclusive	$S^{(e)} = S$
			none-exclusive	$S^{(i)} = S$ *(Not compatible with "multi-room")*
			some-exclusive	$S^{(e)} \neq \emptyset \wedge S^{(i)} \neq \emptyset$
○	teaching	+	no-teacher	$\exists p \in P,\ min\_lecturer_p^P = max\_lecturer_p^P = 0$
			single-teacher	$\exists p \in P,\ min\_lecturer_p^P = max\_lecturer_p^P = 1$
			multi-teacher	$\exists p \in P,\ min\_lecturer_p^P \geq 1 \wedge max\_lecturer_p^P > 1$
		○	session-overlap	$\forall t \in T, \forall s_i, s_j \in d_t^{T,S}, s_i + length_{s_i}^{S} \leq s_j \vee s_i \geq s_j + length_{s_j}^{S}$
			service	*service constraints apply to teachers*
●	attending	○	session-overlap	$\forall g \in G, \forall s_i, s_j \in d_g^{G,S}, s_i + length_{s_i}^{S} \leq s_j \vee$ $s_i \geq s_j + length_{s_j}^{S}$
			sectioning	*student groups must be fixed and pre-assigned to classes*
○	crosscutting	○	calendar	*allowed_slots, forbidden_slots, allowed_grids, ...*
			regularity	*periodic, allowed_grids,same_rooms, different_teachers, ...*
			orchestration	*same_start, different_day, sequenced, no_overlap*
			workload	*compactness, gap, ...*
			logistics	*same_rooms, adjacent_rooms, different_teachers, ...*
			resourcing	*allowed_rooms, required_teachers, ...*

Table 2: A feature model for UTP.

4 Related Work

The design of timetables is a widely studied problem. Given the multitude of situations encountered, simpler, specialized variants of the general problem have been created in order to produce solutions within an acceptable time frame. The best-known variants include ETT (Exam Timetabling) [11,25] which focuses on exams, PE-TT (Post-Enrolment-based Timetabling) [43,51] in which students register for the courses they wish to take, CB-TT (Curriculum-Based Timetabling) [39,5,32], in which students enroll for a curriculum that includes all the courses they have to take, TAP (Tutor Allocation

Problem) [15], which manages the allocation of teachers after the course slots have been set, and HTT (Highschool Timetabling) [33,22], which deals with timetables for high schools.

A timetable design problem is broader than simply scheduling lessons. It depends, for example, on student sectioning [20,6] which consists in dividing students into different groups. But it can also be the starting point for other problems such as BACP [50,18] which seeks to balance teaching periods. Given the difficulty of finding a solution, these ancillary problems are often solved beforehand. Simplification assumptions and resource management differ from problem to problem. Table 3 uses the feature model to compare the scope of the different problems, highlighting the common features and differences.

Although widely studied, the problem of timetable design is often dealt with on an ad-hoc basis. It is a crucial problem in the management of certain institutions which seek above all to produce a solution to their specific problem. This explains the heterogeneity of approaches, making it difficult to evaluate and compare work in the field.

Features	Problems	ETT	CB-TT	PE-TT	HTT	TAP
courses	course-hierarchy		✓			
	event		✓		✓	✓
timing	full-period				✓	
	full-week	✓	✓	✓		
	single-week	✓			✓	
scheduling	no-overlap	✓	✓	✓	✓	
	same-duration	✓	✓	✓	✓	
	modular	✓	✓	✓	✓	
hosting	no-room					NA
	single-room	✓	✓	✓	✓	NA
	multi-room	✓	✓			NA
	room-capacity	✓	✓	✓		NA
	none-exclusive	✓				NA
	all-exclusive	✓	✓	✓	✓	NA
	some-exclusive	✓	✓	✓	✓	NA
teaching	no-teacher		✓	✓		
	single-teacher	✓	✓	✓	✓	✓
	multi-teacher	✓	✓			✓
	session-overlap		✓	✓	✓	✓
	service		✓	✓		
attending	session-overlap	✓	✓		✓	✓
	sectioning		✓		✓	✓
crosscutting	calendar	✓	✓		✓	✓
	regularity	✓	✓	✓		
	orchestration		✓	✓	✓	
	workload	✓	✓	✓	✓	
	logistics	✓				✓
	resourcing		✓	✓	✓	

Table 3: Problem features: a comparison.

The emergence of competitions such as ITC (International Timetabling Competition) has led to the creation of standardized formats, making it easier to compare approaches. ITC-2007, one of the most studied schemas, provides a simplified representation of ETT, PE-TT, and CB-TT. In this schema, the aim is to assign one room and one teacher to each session (single-room,single-teacher). The description of academic courses is carried in CB-TT by the curricula which group the courses together, and in PE-TT by the students (session-overlap). The teachers service is assumed to have already been resolved upstream. A teacher assigned to a course does all the sessions of a course, and sessions are otherwise exclusive (session-overlap). Time is expressed in terms of relative slots, i.e., there is a standard duration of one lesson between 2 slots (no-overlap). Class sessions also all have the same duration (same-duration) and daily slots are repeated in the same pattern every day (synchronous).

The XHSTT-2014 [45,23,21] schema, based on the ITC schema, focuses mainly on modeling timetables for secondary schools. Ancillary problems are solved beforehand: generation of groups, breakdown of rooms, teacher services. In addition to the usual resources (rooms, teacher, students, etc.), it is possible to represent other types of resource (e.g. equipment, vehicles, etc.). However, it is possible to leave out a set of resources on which to make a choice of allocations when solving (single-room,single-teacher). A pre-fit is carried out upstream of the schema to reduce the set of rooms to those authorized according to the size of the groups of students (room-capacity, group). The schema generally contains a single time grid, but there's nothing to stop having several. With this schema, the objective of the solver is to build a typical week (single-week,periodicity). The model proposes a catalog of constraints: hard constraints are interpreted as core constraints, while soft constraints have a violation score to minimize (session-distribution). Constraints can be imposed on resources (resource-distribution).

The ITC-2019 [41,38,30] model focuses specifically on university timetables, more specifically anglo-saxon universities. The ITC-2019 schema addresses scheduling as a combinatorial optimization problem, with a cost function that takes into account 4 criteria. The criteria concern the choice of time slots for sessions, rooms for sessions, violations of soft constraints and the overlap of sessions per student (session overlap). This model takes into account a time horizon of several weeks (full-period,full-week. Timetables are defined as the repetition over a set of weeks (multi-week) of one or more sessions of the same duration starting on specific days of the week at the same predefined time (periodicity). Each room has a penalty score for a session. This has an impact on the choice of room (single-room, exclusive-room). The choice has been made not to represent teachers, nor groups of students. A problem expressed in this model comprises a constraint catalog made up of flexible constraints with a penalty score. The catalog of constraints is used to ensure quality and to express the different needs of the timetable (session-distribution, availability).

The UTP schema [9] has been designed to represent problems in which students enrol on courses. As with the other schemas, simplifications are made. For example, it is assumed that students are divided up into groups beforehand, just like the teachers (the allocation of a teacher to a group and a session is done during the design process). It allows problems to be represented over a modular time horizon and clearly identifies teachers. It also allows resources (rooms, teachers, etc.) to be treated disjunctively or

Features \ Schemas		ITC-2007	ITC-2019	XHSTT-14	UTP
courses	course-hierarchy		✓		✓
	event	✓		✓	✓
timing	full-period	✓	✓		✓
	full-week	✓	✓	✓	✓
	single-week	✓		✓	
scheduling	same-duration	✓	✓	✓	✓
	no-overlap	✓	✓	✓	✓
	modular	✓	✓	✓	✓
hosting	no-room			✓	✓
	single-room	✓	✓	✓	✓
	multi-room				✓
	room-capacity	✓	✓		✓
	all-exclusive	✓	✓	✓	✓
	none-exclusive				✓
	some-exclusive				✓
teaching	no-teacher		NA	✓	✓
	single-teacher	✓	NA	✓	✓
	multi-teacher		NA		✓
	session-overlap		NA	✓	✓
	service		NA		✓
attending	session-overlap		✓	✓	✓
	sectioning			✓	✓
crosscutting	calendar	✓	✓	✓	✓
	regularity	✓	✓	✓	✓
	orchestration	✓	✓	✓	✓
	workload		✓		
	logistics				✓
	resourcing		✓	✓	✓

Table 4: Schema features.

cumulatively according to need. A few changes have been made since [9]. In [9], the problem of groups sectioning is dealt in conjunction with that of designing the timetable. However, a timetable is often designed on the basis of provisional enrolments, as the definitive enrolments are not yet closed. It is therefore not possible to set up the actual groups at such an early stage. It is interesting to be able to dissociate these two problems and, as with the other schemas, sectioning is considered to have been resolved upstream. The `UTP` schema takes as input the list of groups formed. In [9], the time grid is identical whatever the week. In the current version, this can be adapted for a particular day or set of days in the entire time horizon.

Most of the schemas presented above stem from a desire to abstract and generalize a real variant of the problem. Thus, certain assumptions and simplifications are made, limiting the expressiveness of the schema, in particular to express other variants orthogonal to the initial assumptions. By analyzing Table 4, we can cite 3 cases where these simplifying assumptions prevent the representation of other variants: the management of the time horizon, the management of teacher services and the management of resources. As far as time horizon management is concerned, only ITC-2019 and UTP can represent a problem with a time horizon longer than a week. Managing the timetable on a weekly basis is incompatible with institutions where each week is different. With the exception of UTP, no schema takes into account the representation of a teacher's service. They consider that teachers are assigned upstream and cannot be exchanged. However, when several groups follow the same course and several teachers are involved, this removes flexibility and prevents certain solutions that could be of high quality from being achieved. Finally, the management of resources also differs from one schema to another. XHSTT and UTP allow teachers to be represented as such, whereas the ITC models do not explicitly include them. In addition, whether for rooms or teachers, the various schemas, apart from UTP, do not allow the resource to be shared over several sessions. For example, it is not possible to represent a problem where one teacher supervises several practical sessions. Nor is it possible to represent a problem in which a session must be hosted in several rooms (adjacent or not). Only UTP can represent problems in which the resources are disjunctive or cumulative.

Competitions are regularly organized [29] and provide an opportunity to make available a set of real or fictitious instances, enabling any new algorithm to be compared with existing approaches. Whether for simulation or comparison purposes, it is useful to have a means of generating new instances. Only the ITC-2007 schema has an instance generator. Developed in 2008, this generator was improved in 2010 and again in 2022 to produce more realistic instances and better cover the range of possible configurations (available on [2]).

5 Instance Generator and Experiments

This section introduces a generator of pseudo-random UTP instances and reports on experiments carried out with three models, namely, CP, ASP and MIP. The objective is to assess the scalability of the models and their applicability to real-life instances. The complete list of instances and the models may be found in Appendix.

5.1 Instance Generator

To generate a UTP instance involves generating a course structure, groups of students, teacher services, and rules. All those generators can be configured thanks to XML files to select the features we want our instance to fit in.

In our generator, we define curricula associated with faculty departments. Curricula enable us to associate a set of courses with a set of students and a set of teachers. Each department is associated with a set of courses, teachers and specific rooms (*i.e.* rooms that can be used only by courses of the department). The courses are divided into several parts. The number of parts usually varies between 1 (only lectures) and 4 (lectures, tutorials, practices and evaluations).

Student sectioning is the problem of assigning students to groups. The UTP schema takes groups, rather than individual students, necessitating that the generator supply groups. The input of the generator is the number of students enrolled in a curriculum. A CSP model is used by the generator to create groups. The different sizes of groups

name	$\|R\|$	$\|T\|$	$\|U\|$	$\|S\|$	#ru	CP		ASP		CASP		MIP	
						BT	ST	BT	ST	BT	ST	BT	ST
gi8201	82	4	33	118	23	0.86	1.86	15.71	0.08	0.30	0.23	0.17	0.26
gi5301	99	96	55	139	33	0.85	483.59	68.70	0.36	0.40	0.12	0.22	0.49
gi4389	65	6	174	662	33	1.47	4.62	693.40	210.58	14.91	4.93	7.66	10.23
gi5567	94	94	1770	6180	562	4.75	381.78			122.89	3548.58		
gi2767	92	21	501	2003	78	2.02	30.48			128.21	57.11	130.51	7995.32
real	117	183	768	2625	520	2.21	8.71			219.97	411.79	-	-

Table 5: Selected list of instances. $|R|$ is the number of rooms, $|T|$ the number of teachers, $|U|$ the number of students, $|S|$ the number of sessions, #ru the number of rules; BT is the building time (s) and ST the solving time (s).

(lecture, tutorial, practice) should be given. It is possible to change the size of a specific group for a curriculum (e.g., a specific curriculum with group sizes different from the standard ones). The sectioning CSP can create groups with a fixed size, or create courses with a limited number of groups, to fix the total number of hours. Teaching service is a problem where we know how many hours a teacher has to do, how many hours each part of a class lasts, and we want to assign each teacher with a number of classes in each part. This will give us all the course parts a teacher has to teach. Note that we just know how many classes a teacher is assigned to, not to which class, which is another problem known as the tutor allocation problem. The generator uses a CSP model to tackle this problem.

There are various rules, often used together with some being more common. We defined three rule packs: 1) light: some classes have `same_rooms`, `same_teachers`, `periodic`, and a `sequenced` rule between two parts of the course; 2) medium: all classes have `same_rooms` and `same_teachers`, with some also having `periodic` and `sequenced`; 3) heavy: like medium, but with additional `same_teachers` and `same_start` rules for classes in the same part.

5.2 Instances and Results

We carried out experiments on pseudo-random instances built with our generator and on a real-life instance from our Computer Sciences department. The pseudo-random instances are listed in Appendix 6 and may be downloaded from [4]. A selected subset is given in Table 5.

The generated instances were built by varying the number of rooms, groups and sessions. All are single-room and single-teacher and uses the "medium" rules pack, meaning that all classes have `same_rooms` and `same_teachers`, and some have `periodic` and a `sequenced` constraint between 2 different parts of the same course. The real-life instance consists of the 3 years of bachelor and the 2 years of master in Computer Sciences at the University of Angers in 2023. The instance is reduced only to courses that occur at the first time period of each curriculum.

The experiments were performed on a computer with a processor Intel-Xeon E7-4850 v4, 2.1 GHz, 40MB of cache. The CP solver is Choco-solver [49] 4.10.12. The ASP solver is Clingo [24] 5.6.2. The CASP solver is Clingcon [46] 5.2.0. The MIP solver is Gurobi Optimizer [28] 10.0.3. The building (BT) and solving (ST) times can be found in Appendix 6. Table 5 shows selected instances: an instance where all solvers performed well (gi8201) and the worst-case instances in run-time for CP (gi5301), ASP (gi4389), CASP (gi5567) and MIP (gi2767). Those instances show the limits of the solvers. In particular, the more sessions there are, the more difficult it is for ASP and MIP to solve instances to completion. Some instances could not be solved by ASP as it may need more than 80GB when MIP instances use up to 15GB, and CP and CASP only need 8GB. Some instances could not be solved by MIP due to time outs with a time limit set to 5 hours.

6 Conclusion

In this article, we focused on a class of timetabling problems (UTP), proposing a framework that can adapt to different types of institutions, whether they operate like high schools or universities, and to account for regular classes, exams, meetings, or special events. Our current work addresses several aspects. Firstly, we aim to experiment on larger real-world instances and are developing a set of software applications for this purpose. We are also working on incorporating soft constraints and priorities to propose a solution in cases where there is no solution that satisfies all expressed constraints. Finally, we are working on timetable revisions to accommodate unforeseen events such as teacher absences or room unavailability.

Appendix A – Core Computational Model

UTP instances are cast as hard constraint satisfaction problems. We present here a formal specification of the constraint-based model for the entity schema. This core model only formulates the built-in constraints that apply to any instance, leaving out the constraints generated from instance-specific rules.

Table 6 lists the core decision variables of the model and includes auxiliary variables for notational convenience. All, except temporal variables, are cast as set variables.

$x_s^{S,H}$	$\in H$	the start time assigned to session s
$x_s^{S,R}$	$\subseteq R$	the set of rooms assigned to session s
$x_s^{S,T}$	$\subseteq T$	the set of lecturers assigned to session s
$x_s^{S,W}$	$\in W$	(auxiliary variable) the week of the start time assigned to session s
$x_s^{S,D}$	$\in D$	(auxiliary variable) the weekday of the start time assigned to session s
$x_s^{S,M}$	$\in M$	(auxiliary variable) the daily slot of the start time assigned to session s

Table 6: Core and auxiliary decision variables.

Table 7 formulates the core constraints of the model. Note that some statements reify primitive constraints (e.g., set memberhsip) as implicit pseudo-boolean variables.

Constraint (1) establishes the relationship between the start time of a session and its start points on the 3 time scales. (2) restricts the start time of a session to the allowed start points. Constraint (3) ensures that every session starts and ends on the same day. Constraint (4) sequences the sessions of a class based on ranks. (5) models the sets of possible and required rooms for a session as set inclusion constraints. (6) is the counterpart for teachers. (7) and (8) set the bounds on the number of rooms and teachers per session. (9) models the service constraint for each teacher measured in number of sessions. Constraints (10) and (11) model the cumulative capacity of rooms and address the 3 hosting scenarios discussed in Section 2.1 (multi-room exclusive, single-room exclusive and single-room inclusive sessions). (10) ensures any (single- or multi-room) exclusive session allocated to a room virtually fulfills the room capacity and hence has exclusive use of it. Otherwise, the total headcount of the inclusive session(s) occupying the room at any time must not exceed its capacity. Constraint (11) models the capacity demand of each session, be it single- or multi-rooms, and ensures its allocated room(s) meet the demand.

$\forall s \in S$	$x_s^{S,H} = x_s^{S,M} + m * (x_s^{S,D} - 1) + m * d * (x_s^{S,W} - 1)$	(1)
$\forall s \in S$	$x_s^{S,H} \in d_s^{S,H}$	(2)
$\forall s \in S$	$x_s^{S,D} + (x_s^{S,W} - 1) * d = (x_s^{S,H} + length_s^S) \div m$	(3)
$\forall c \in K,\ \forall s_1, s_2 \in d_c^{K,S}$	$rank_{s_1}^S < rank_{s_2}^S \rightarrow (x_{s_1}^{S,H} + length_{s_1}^S) \leq x_{s_2}^{S,H}$	(4)
$\forall s \in S$	$\underline{d}_s^{S,R} \subseteq x_s^{S,R} \subseteq \overline{d}_s^{S,R}$	(5)
$\forall s \in S$	$\underline{d}_s^{S,T} \subseteq x_s^{S,T} \subseteq \overline{d}_s^{S,T}$	(6)
$\forall p \in P, \forall s \in d_p^{P,S}$	$min\_rooms_p^P \leq \lvert x_s^{S,R} \rvert \leq max\_rooms_p^P$	(7)
$\forall p \in P, \forall s \in d_p^{P,S}$	$min\_lecturer_p^P \leq \lvert x_s^{S,T} \rvert \leq max\_lecturer_p^P$	(8)
$\forall t \in T,\ \forall p \in P$	$service_{l,p}^{T \times P} = \sum_{s \in d_p^{P,S}} (l \in x_s^{S,T})$	(9)
$\forall r \in R, \forall h \in H$	$capacity_r^R \geq \sum_{s \in S^{(i)}} (r \in x_s^{S,R}) * (h = x_s^{S,H}) * (size_{d_s^{S,K}}^K) +$ $\sum_{s \in S^{(e)}} (r \in x_s^{S,R}) * (h = x_s^{S,H}) * (capacity_r^R)$	(10)
$\forall s \in S, \forall k \in d_s^{S,K}$	$d_k^{K,R} \neq \emptyset \rightarrow size_k^K \leq \sum_{r \in x_s^{S,R}} capacity_r^R$	(11)

Table 7: Core constraints.

Appendix B – Rules Syntax and Constraint Predicate Catalog

Table 8 provides the syntax of the rules language: e-maps, constraints, queries and rules.

(e_i, S_i)	e-map mapping entity e_i to the set of sessions S_i
$F = E \times 2^S$	the domain of e-maps
$c((e_1, S_1), .., (e_n, S_n), p_1, .., p_m)$	constraint of predicate c, arity n, parameters $p_1, .., p_m$ and e-map arguments $(e_1, S_1), .., (e_n, S_n) \in F^n$
$O = 1.. \max_{s \in S} rank_s^S$	the range of session ranks
$\mathcal{L}^* = \mathcal{L} \cup \{E\} \cup \{\{e\} \mid e \in E\}$	the set of labels
$Q = \cup_{n \geq 1} (\mathcal{E} \times \mathcal{L}^* \times 2^O)^n$	the language of queries
$c\langle Q, p_1, \ldots, p_m \rangle$	rule of predicate c, query Q and parameters $p_1, \ldots p_m$ - c predicate of arity n and number of parameters m - Q query sized to extract n sets of e-maps - $p_1, \ldots p_m$ values for the parameters of c

Table 8: Predicates, constraints, queries and rules.

Table 9 lists and describes the predicates of the catalog.

Predicate	**Description**
adjacent_rooms	Sessions must be adjacent in the given room(s)
allowed_grids	Sessions may only start in the given time grid(s)
allowed_rooms	Sessions may only be hosted in the given room(s)
allowed_slots	Sessions may only run in the given time slots
allowed_teachers	Sessions may only be taught by the given teacher(s)
assign_rooms	Sessions are hosted in the given room(s)
assign_start	Sessions start at the given time
assign_teachers	Sessions are taught by the given teacher(s)
compactness	The sessions makespan is bounded
different_daily_start	Sessions start on different daily slots
different_day	Sessions start on different days
different_rooms	Sessions are hosted in different rooms
different_starts	Sessions start at different times
different_teachers	Sessions are taught by different teachers
different_week	Sessions start on different week
different_weekday	Sessions start on different weekday
different_weekly_start	Sessions start on different weekly time points
forbidden_rooms	Sessions cannot be hosted in the given room(s)
forbidden_slots	Sessions cannot run in the given time slots
forbidden_teachers	Sessions cannot be taught by the given teacher(s)
gap	Gaps between sessions are bounded
no_overlap	Sessions in the given set cannot overlap
pairwise_no_overlap	Sessions cannot overlap if in different sets
periodic	Sessions are periodic
required_rooms	Sessions must be hosted in the given room(s)
required_teachers	Sessions must be taught by the given teacher(s)
same_daily_start	Sessions start on the same daily slot
same_day	Sessions start on the same day
same_rooms	Sessions are hosted in the same room(s)
same_start	Sessions start at the same time
same_teachers	Sessions are taught by the same teacher(s)
same_weekday	Sessions start on the same weekday
same_weekly_start	Sessions start on the same weekly time point
same_week	Sessions start on the same week
sequenced	Sessions run sequentially
sessions_workload	The number of sessions per time frame is bounded
times_workload	The total duration of sessions per time frame is bounded

Table 9: Catalog of UTP constraint predicates.

Table 10 provides the semantics of each predicate once the scope is restricted to a tuple of sets of sessions (obtained after joining a solution and a constraint built with the predicate). Given a n-ary predicate c accepting m parameters ($m \geq 0$) and given a n-uple $(S'_1, \ldots, S'_n) \in S^n$, we provide the semantics for $c(S'_1, \ldots, S'_n, p_1, \ldots, p_m)$ which denotes the evaluation of the predicate on the tuple.

Predicate **Semantics**	**Parameters**
adjacent_rooms(S',K$_1$,...,K$_n$)	$K_i \subseteq R, i : 1..n$
$\forall s_1, s_2 \in S', x^{S,H}_{s_1} = x^{S,H}_{s_2}$ $\forall i \in \{1..n\} : C_i = \{x^{S,R}_s \mid s \in S', x^{S,R}_s \in K_i\} \wedge$ $\forall r, r' \in C_i : \exists path(r, r') \wedge \lvert\{C_i \mid C_i \neq \emptyset, i = 1..n\}\rvert \leq \sigma$	(1)
allowed_grids(S',G)	$G \in (W' \times D' \times M')^n, n \in \mathbb{N}^*,$ $S' \subseteq S, W' \subseteq W, D' \subseteq D, M' \subseteq M$
$\forall s \in S', \exists k \in \{1..n\} : x^{S,W}_s \in W'_k \wedge x^{S,D}_s \in D'_k \wedge x^{S,M}_s \in M'_k$	(2)
allowed_rooms(S',R')	$S' \subseteq S, R' \subseteq R$
$\forall s \in S', x^{S,R}_s \subseteq R'$	(3)
allowed_slots(S',H')	$H' \subseteq H,$
$\forall s \in S' [x^{S,H}_s, x^{S,H}_s + length^S_s] \subseteq H'$	(4)
allowed_teachers((e,T'),R')	$S' \subseteq S, T' \subseteq T$
$\forall s \in S', x^{S,T}_s \subseteq T'$	(5)
assign_rooms(S',R')	$R' \subseteq R$
$\forall s \in S', x^{S,R}_s = R'$	(6)
assign_start(S',H')	$h \in H$
$\forall s \in S', x^{S,H}_s = h$	(7)
assign_teachers S',T')	$T' \subseteq T$
$\forall s \in S', x^{S,T}_s = T'$	(8)
compactness(S',σ)	$\sigma \in 0..\lvert H'\rvert$
$\forall d \in D : \exists S'' = \{s \in S' : x^{S,D}_s\} \wedge$ $((\max\limits_{s \in S''}(x^{S,H}_s + length^S_s) - \min\limits_{s \in S'}(x^{S,H}_s)) - \sum_{s \in S''} length^S_s))/(\lvert S''\rvert - 1) \leq \sigma$	(9)
different_daily_start(S')	
$\forall s_1, s_2 \in S', x^{S,M}_{s_1} \neq x^{S,M}_{s_2}$	(10)
different_day(S')	
$\forall s_1, s_2 \in S', x^{S,D}_{s_1} \neq x^{S,D}_{s_2} \vee x^{S,W}_{s_1} \neq x^{S,W}_{s_2}$	(11)
different_rooms(S')	
$\forall s_1, s_2 \in S', x^{S,R}_{s_1} \cap x^{S,R}_{s_2} = \emptyset$	(12)
different_starts(S')	
$\forall s_1, s_2 \in S', x^{S,H}_{s_1} \neq x^{S,H}_{s_2}$	(13)
different_teachers(S')	
$\forall s_1, s_2 \in S', x^{S,T}_{s_1} \cap x^{S,T}_{s_2} = \emptyset$	(14)
different_week(S')	
$\forall s_1, s_2 \in S', x^{S,W}_{s_1} \neq x^{S,W}_{s_2}$	(15)
different_weekday(S')	

$\forall s_1, s_2 \in S', x^{S,D}_{s_1} \neq x^{S,D}_{s_2}$ (16)

different_weekly_start(S')

$\forall s_1, s_2 \in S', x^{S,M}_{s_1} \neq x^{S,M}_{s_2} \vee x^{S,D}_{s_1} \neq x^{S,D}_{s_2}$ (17)

forbidden_rooms(S',R') $R' \subseteq R$

$\forall s \in S', x^{S,R}_{s} \subseteq R \setminus R'$ (18)

forbidden_slots(S',H') $H' \subseteq H$

$\forall s \in S' [x^{S,H}_{s}, x^{S,H}_{s} + length^{S}_{s}[\cap H' = \emptyset$ (19)

forbidden_teachers(S',T') $T' \subseteq T$

$\forall s \in S', x^{S,T}_{s} \subseteq T \setminus T'$ (20)

min_slot_gap(S',σ_{min}) $\sigma_{min} \in 0..|H|,$

$\exists! \pi : S' \rightarrow [[S']] : x^{S,H}_{\pi^{-1}(i)} < x^{S,H}_{\pi^{-1}(j)} \quad {\scriptstyle (1 \leq i < j \leq |S'|)}$

$\forall i \in 1..|S'| - 1, x^{S,H}_{\pi^{-1}(i+1)} - (x^{S,H}_{\pi^{-1}(i)} + length^{S}_{\pi^{-1}(i)}) \geq \sigma_{min}$ (21)

min_day_gap(S',σ_{min}) $\sigma_{min} \in 0..|D| * |W|,$

$\exists! \pi : S' \rightarrow [[S']] : x^{S,H}_{\pi^{-1}(i)} < x^{S,H}_{\pi^{-1}(j)} \quad {\scriptstyle (1 \leq i < j \leq |S'|)}$

$\wedge \forall i \in 1..|S'| - 1, x^{S,D}_{\pi^{-1}(i+1)} - (x^{S,D}_{\pi^{-1}(i)}) \geq \sigma_{min}$ (22)

min_week_gap(S',σ_{min}) $\sigma_{min} \in 0..|W|,$

$\exists! \pi : S' \rightarrow [[S']] : x^{S,H}_{\pi^{-1}(i)} < x^{S,H}_{\pi^{-1}(j)} \quad {\scriptstyle (1 \leq i < j \leq |S'|)}$

$\wedge \forall i \in 1..|S'| - 1, x^{S,W}_{\pi^{-1}(i+1)} - (x^{S,W}_{\pi^{-1}(i)}) \geq \sigma_{min}$ (23)

max_slot_gap(S',σ_{max}) $\sigma_{max} \in 0..|H|,$

$s_1 = \min_{s \in S'}(x^{S,H}_{s}), s_2 = \max_{s \in S'}(x^{S,H}_{s} + length^{S}_{s}), x^{S,H}_{s_2} - (x^{S,H}_{s_1} + length^{S}_{s}) \leq \sigma_{max}$ (24)

max_day_gap(S',σ_{max}) $\sigma_{max} \in 0..|D| * |W|,$

$s_1 = \min_{s \in S'}(x^{S,H}_{s}), s_2 = \max_{s \in S'}(x^{S,H}_{s}), x^{S,D}_{s_2} - x^{S,D}_{s_1} \leq \sigma_{max}$ (25)

max_week_gap(S',σ_{max}) $\sigma_{max} \in 0..|W|,$

$s_1 = \min_{s \in S'}(x^{S,H}_{s}), s_2 = \max_{s \in S'}(x^{S,H}_{s}), x^{S,W}_{s_2} - x^{S,W}_{s_1} \leq \sigma_{max}$ (26)

last_first_slot_gap$(S'_1 \ldots ,S'_n,\sigma_{min},\sigma_{max})$ $\sigma_{min}, \sigma_{max} \in 0..|H|,$

$\forall i \in 1..n-1, \; s_i = \max_{s \in S_i}(x^{S,H}_{s} + length^{S}_{s}), \; s_{i+1} = \min_{s \in S_{i+1}}(x^{S,H}_{s})$

$\sigma_{min} \leq s_{i+1} - s_i \leq \sigma_{max}$ (27)

last_first_day_gap$(S'_1 \ldots ,S'_n,\sigma_{min},\sigma_{max})$ $\sigma_{min}, \sigma_{max} \in 0..|H|,$

$\forall i \in 1..n-1, \; s_i = \max_{s \in S_i}(x^{S,D}_{s}), \; s_{i+1} = \min_{s \in S_{i+1}}(x^{S,D}_{s})$

$\sigma_{min} \leq s_{i+1} - s_i \leq \sigma_{max}$ (28)

last_first_week_gap$(S'_1 \ldots ,S'_n,\sigma_{min},\sigma_{max})$ $\sigma_{min}, \sigma_{max} \in 0..|H|,$

$\forall i \in 1..n-1, \; s_i = \max_{s \in S_i}(x^{S,W}_{s}), \; s_{i+1} = \min_{s \in S_{i+1}}(x^{S,W}_{s})$

$\sigma_{min} \leq s_{i+1} - s_i \leq \sigma_{max}$ (29)

first_last_slot_gap$(S'_1 \ldots ,S'_n,\sigma_{min},\sigma_{max})$ $\sigma_{min}, \sigma_{max} \in 0..|H|,$

$\forall i \in 1..n-1, \; s_i = \min_{s \in S_i}(x^{S,H}_{s} + length^{S}_{s}), \; s_{i+1} = \max_{s \in S_{i+1}}(x^{S,H}_{s})$

$$\sigma_{min} \leq s_{i+1} - s_i \leq \sigma_{max} \tag{30}$$

first_last_day_gap$(S'_1 \ldots, S'_n, \sigma_{min}, \sigma_{max})$ $\quad \sigma_{min}, \sigma_{max} \in 0..|H|,$

$$\forall i \in 1..n-1,\ s_i = \min_{s \in S_i}(x_s^{S,D}),\ s_{i+1} = \max_{s \in S_{i+1}}(x_s^{S,D})$$
$$\sigma_{min} \leq s_{i+1} - s_i \leq \sigma_{max} \tag{31}$$

first_last_week_gap$(S'_1 \ldots, S'_n, \sigma_{min}, \sigma_{max})$ $\quad \sigma_{min}, \sigma_{max} \in 0..|H|,$

$$\forall i \in 1..n-1,\ s_i = \min_{s \in S_i}(x_s^{S,W}),\ s_{i+1} = \max_{s \in S_{i+1}}(x_s^{S,W})$$
$$\sigma_{min} \leq s_{i+1} - s_i \leq \sigma_{max} \tag{32}$$

no_overlap(S')

$$\bigwedge_{\substack{s_1, s_2 \in S' \\ s_1 \neq s_2}} (x_{s_1}^{S,H} + length_{s_1}^S \leq x_{s_2}^{S,H}) \vee (x_{s_2}^{S,H} + length_{s_2}^S \leq x_{s_1}^{S,H}) \tag{33}$$

pairwise_no_overlap(S'_1, S'_2)

$$\bigwedge_{\substack{s_1 \in S'_1, s_2 \in S'_2 \\ s_1 \neq s_2}} (x_{s_1}^{S,H} + length_{s_1}^S \leq x_{s_2}^{S,H}) \vee (x_{s_2}^{S,H} + length_{s_2}^S \leq x_{s_1}^{S,H}) \tag{34}$$

periodic(S', n) $\quad n \in \mathbb{N}$

$$\exists \pi : S' \longrightarrow 1..|S'|, \forall i \in 1..|S'|-1, x_{\pi^{-1}(i)}^{S,H} + n = x_{\pi^{-1}(i+1)}^{S,H} \tag{35}$$

required_rooms(S', R') $\quad R' \subseteq R$

$$\forall s \in S',\ R' \subseteq x_s^{S,R} \tag{36}$$

required_teachers(S', T', Δ^*) $\quad T' \subseteq T, \Delta = \{\forall t \in T' \mid \delta_{1,t}, \delta_{2,t}\},$ $\forall t \in T', \forall i \in \{1,2\},\ \delta_{i,t} \in \mathbb{N}$

$$\forall s \in S',\ (x_s^{S,T} \subseteq T' \wedge \forall t \in T',\ \delta_{1,t} \leq \sum_{s \in S'} (t \in x_s^{S,T}) \leq \delta_{2,t}) \tag{37}$$

same_daily_start (S')

$$\forall s_1, s_2 \in S', x_{s_1}^{S,M} = x_{s_2}^{S,M} \tag{38}$$

same_day(S')

$$\forall s_1, s_2 \in S', x_{s_1}^{S,D} = x_{s_2}^{S,D} \wedge x_{s_1}^{S,W} = x_{s_2}^{S,W} \tag{39}$$

same_rooms(S')

$$\forall s_1, s_2 \in S', x_{s_1}^{S,R} = x_{s_2}^{S,R} \tag{40}$$

same_start(S')

$$\forall s_1, s_2 \in S', x_{s_1}^{S,H} = x_{s_2}^{S,H} \tag{41}$$

same_teachers(S')

$$\forall s_1, s_2 \in S', x_{s_1}^{S,T} = x_{s_2}^{S,T} \tag{42}$$

same_week(S')

$$\forall s_1, s_2 \in S', x_{s_1}^{S,W} = x_{s_2}^{S,W} \tag{43}$$

same_weekday(S')

$$\forall s_1, s_2 \in S', x_{s_1}^{S,D} = x_{s_2}^{S,D} \tag{44}$$

same_weekly_start(S')

$$\forall s_1, s_2 \in S', x_{s_1}^{S,M} = x_{s_2}^{S,M} \wedge x_{s_1}^{S,D} = x_{s_2}^{S,D} \tag{45}$$

sequenced$(S'_1 \ldots, S'_n)$

$$\forall s_i \in S'_i, \forall s_{i+1} \in S'_{i+1},\ x_{s_{i+1}}^{S,H} \geq x_{s_i}^{S,H} + length_{s_i}^S \tag{46}$$

time_workload(S', w_1, w_2) $\quad w_1, w_2 \in H \cup \{0\}$

$$\forall d \in D,\ w_1 \leq \sum_{s \in S'} length_s^S \times (x_s^{S,D} = d) \leq w_2 \tag{47}$$

session_workload(S',w_1, w_2)	$w_1, w_2 \in 0..\|S\|$
$\forall d \in D,\ w_1 \leq \|\{s \in S' : x_s^{S,D} = d\}\| \leq w_2$	(48)

Table 10: Semantics of UTP constraint predicates

Appendix C – Models

The CP model (see table 11 in appendix 6) for UTP is based on decisions variables listed in table 6. For the constraint related to teacher service (requiredTeacher), we using the global cardinality constraint (gcc), which enables element counting. It is also feasible to add counting constraints to better distribute the workload or limit the number of hours per day for a room, aiming to achieve more robust solutions. For conditional constraints, we can apply the reification pattern (checking the feasibility and consumption of potential values and use possible sink state). For constraints of equality, we employ the global constraint all_equal, which ensures for input variables, all have the same value (the implemented propagator is similar to gcc). For the no_overlap constraint and for the main room usage constraint, we employ the global cumulative constraint to ensure that, when sessions utilizing a resource, its maximum capacity is not exceeded, thereby preventing multiple class sessions from overlapping. For difference/disjoint constraints, we used the global n-ary constraints all_different for integer variables and all_disjoint for set variables.

ASP [8] is a form of declarative programming for solving difficult search problems. It operates by defining problems in terms of rules and constraints, then computing the “answer sets” which are collections of assumptions that satisfy rules and constraints without contradiction. The programmer specifies the desired properties of the solution in a high-level language, and the ASP system automatically searches for all solutions that meet these criteria, making it a powerful tool for knowledge representation and reasoning tasks. ASP has been used to address timetabling problems proposed in the ITC-2007 competition [7]. The ASP program we propose for UTP (see appendix 6 for the full program) has been developed with clingo - an ASP solver - and clingcon - a constraint answer set programing solver.

In the state of the art [47,35], mixed-integer linear programming (MIP) models are usually used to solve timetable scheduling problems. In the literature, MIP models are presented in a pseudo-Boolean format, where time is represented in a time-indexed representation. This implies that time is discretized into intervals from 0 to 1 for each time slot. For the UTP problem, where the time horizon is extended, classical representations are not very efficient. Indeed, time-indexed representations exhibit exponential growth. In other scheduling problems such as RCPCSP, representations can be time-indexed, or as in CP in continuous time slots (integer value), or even an event-based approach. Continuous time variables in MIP offer advantages in terms of variable economy, but they require techniques such as big-M to express disjunctions. In some articles [34], it has been demonstrated that event-driven approaches are more effective than continuous or time-indexed approaches for extended time horizons. Here, we present a MIP program for UTP. The program will be used in our experimental study at Section 5(see Appendix 6 for the full program).

C.1 – CP model

$\forall s \in S$
$$x_s^{S,R} \subseteq d^{P,R}_{d_s^{S,P}} \tag{1}$$

$\forall s \in S$
$$x_s^{S,T} \subseteq d^{P,T}_{d_s^{S,P}} \tag{2}$$

$\forall p \in P,\ \forall s \in d_p^{P,S}$
$$min\_rooms_p^P \leq |x_s^{S,R}| \leq max\_rooms_p^P \tag{3}$$

$\forall p \in P,\ \forall s \in d_p^{P,S}$
$$min\_lecturer_p^P \leq |x_s^{S,T}| \leq max\_lecturer_p^P \tag{4}$$

$\forall p \in P,\ \forall k \in K$
$$size_k^K \leq \sum_{r \in R} (r \in x_s^{S,R}) * capacity_r^R \tag{5}$$

$\forall s \in S :$
$$x_s^{S,M} + (x_s^{S,D} - 1) * |M| + (x_s^{S,W} - 1) * |D| * |M| = x_s^{S,H} \tag{6}$$

$\forall r \in R :$
$$cumulative(\{\forall s \in S \mid (r \in x_s^{S,R}) * x_s^{S,H}\}, \{\forall s \in S \mid length_s^S\}, 1) \tag{7}$$

$\forall p \in P,\ \forall r \in d_p^{P,R} :$
$$cumulative(\{\forall s \in d_p^{P,S} \mid (r \in x_s^{S,R}) * x_s^{S,H}\}, \{\forall s \in d_p^{P,S} \mid length_s^S\}, 1) \tag{8}$$

$\forall p \in P :$
$$gcc(\{\forall s \in d_p^{P,S} \mid x_s^{S,R}\}, \{\forall s \in d_p^{P,S} \mid length_s^S\}, 1) \tag{9}$$

$forbidden\_period(S') = \forall s \in S'$
$$(x_s^{S,H} + length_s^S \leq h_1) \vee (x_s^{S,H} \geq h_2) \tag{10}$$

$same\_weekday(S') = \forall s_1, s_2 \in S', s_1 < s_2$
$$x_{s_1}^{S,D} = x_{s_2}^{S,D} \tag{11}$$

$same\_rooms(S') = \forall s_1, s_2 \in S', s_1 < s_2$
$$x_{s_1}^{S,R} = x_{s_2}^{S,R} \tag{12}$$

$different\_rooms(S') =$
$$all\_disjoint(\{\forall s \in S' \mid x_s^{S,R}\}) \tag{13}$$

$different\_starts(S') =$
$$all\_different(\{\forall s \in S' \mid x_s^{S,H}\}) \tag{14}$$

$sequenced(S_1, S_2) = \forall s_1 \in S_1,\ \forall s_2 \in S_2$
$$x_{s_1}^{S,H} + length_{s_1}^S \leq x_{s_2}^{S,H} \tag{15}$$

$no\_overlap(S') =$
$$\text{cumulative}(\{\forall s \in S' \mid x_s^{S,H}\}, \{\forall s \in S' \mid length_s^S\}, 1) \tag{16}$$

Table 11: Constraints and predicates of the CP model.

C.2 – ASP model

The ASP model is split in two parts: the first part lists the facts with a small example (Listing 10.1), the second part is the declaration of rules to solve the given UTP problem (Listing 10.2).

```
weeks(4). days(12). slot_per_day(1440).grid(480,90,7).
courses(1). parts(2).classes(3). sessions(12).
room(Salle-1,40). room(Salle-2,20). room(Salle-3,20).
course(math). teacher(teacher1).teacher(teacher2).
part(math-CM,12,120,5,12,2,1). class(math-CM-1,80).
    course_part(math,math-CM).
part_class(math-CM,math-CM-1). class_sessions(math-CM-1,1..12).
part_teacher(math-CM,(teacher-1;teacher-2)).
part_room(math-CM,(salle-1;salle-2;salle-3)).
part_days(math-CM,1..5). part_weeks(math-CM,1..12).
part_slots(math-CM, (480;570;660;750;840;930)).
part_grids("cours-1-pCM",1,1,6).
group(group-1,20).group(group-2,20).
    class_group(math-CM-1,(group-1;group-2)).
session_duration(S,D) :- session(S),session_part(S,P),
    part_grids(P,_,D,_).
session_group(S,G) :- class_sessions(C,S),class_group(C,G).
session_part(S,P) :- session(S), class_session(C,S),
    part_class(P,C).
session_teacher(S,T) :- session(S), session_part(S,P),
    part_teacher(P,T).
session_room(S,R) :- session(S), session_part(S,P),
    part_room(P,R).
sequenced(3,(6;7)).
periodic(S1,S2,7200) :- session(S1),session(S2),S2 = S1+1 .
disjunctive_room((1..12),R):- room(R,_).
disjunctive_teacher((1..12),T):- teacher(T).
```

Listing 10.1: ASP facts

```
1{assigned(S,SL) : session_part(S,P), part_slot(P,SL)}1 :-
    session(S).
nrPositionRoom(S,1..N) :- session(S), nrRoomMax(S,N).
nrRoomMax(S,N) :- session(S), session_part(S,P),
    part(P,_,_,_,_,N,_), N > 1.
sessionRoomFix(S) :- session(S), not nrRoomMax(S,_).
partRoomFix(P) :- part_sessions(P,S), sessionRoomFix(S).
partRoomMulti(P) :- part_sessions(P,S), nrRoomMax(S,N).
K{assignedrk(S,SL,I) : nrPositionRoom(S,I)}K :- assigned(S,SL),
    nrRoomMax(S,K).
1{assignedr(S,SL,R,K) : session_room(S,R)}1 :-
    assignedrk(S,SL,K).
1{assignedr(S,SL,R,1) : session_room(S,R)}1 :- assigned(S,SL),
    sessionRoomFix(S).
1{assigned(S,SL) : session_part(S,P), part_slot(P,SL)}1 :-
    session(S).
nrPositionTeacher(S,1..N) :- session(S), nrteacherMax(S,N).
nrteacherMax(S,N) :- session(S), session_part(S,P),
    part(P,_,_,_,_,_,N), N > 1.
sessionTeacherFix(S) :- session(S), not nrteacherMax(S,_).
partTeacherFix(P) :- part_sessions(P,S), sessionTeacherFix(S).
partTeacherMulti(P) :- part_sessions(P,S), nrteacherMax(S,N).
K{assignedtk(S,SL,I) : nrPositionTeacher(S,I)}K :-
    assigned(S,SL), nrteacherMax(S,K).
:- assignedt(S,_,T,K2), assignedt(S,_,T,K1), K1 != K2.

:- not {assignedt(S,SL,T,K) : session(S),
    disjunctive_teacher(T,S), nrPositionTeacher(S,K) } 1,
    slots(SL), teacher(T).
:- not {assignedt(S,SL,T,1) : session(S),
    disjunctive_teacher(T,S) } 1, slots(SL), teacher(T).
:- nrRoomMax(S,_), session_class(S,C), class_headcount(C,N), N
    > #sum{V:assignedr(S,_,R,_),room(R,V)}.
:- assignedr(S,SL,R,1), room(R,C1),session_class(S,C),
    class_headcount(C,N), N > C1.
:- not {assigned(S,SL) : session(S),disjunctive_group(S,G)}1,
    group(G,_), slots(SL).
:- not {assignedr(S,SL,R,K) : session(S),
    disjunctive_room(R,S), nrPositionRoom(S,K) } 1, slots(SL),
    room(R,_).
:- not {assignedr(S,SL,R,1) : session(S), disjunctive_room(R,S)
    } 1, slots(SL), room(R,_).
:- assignedr(S,_,R,K2), assignedr(S,_,R,K1), K1 != K2.
:- periodic(S1,S2,N), assigned(S1,SL1), assigned(S2,SL2), SL2
    != SL1+N.
:- sequenced(S1,S2), assigned(S1,SL1), session_part(S1,P),
    part_grids(P,_,N,_), assigned(S2,SL2), SL1+N > SL2.
sequenced(S1,S2) :- part(P,_,_,_,_,_,_), part_class(P,C),
    class_sessions(C,S1), class_sessions(C,S2), S1+1 = S2.
```

```
:- same_slot(S1,S2), assigned(S1,SL1), assigned(S2,SL2), SL1 !=
    SL2.
:- same_teachers(S1,S2), assignedt(S1,_,T1,K),
    assignedt(S2,_,T2,K), T1 != T2.
:- same_rooms(S1,S2), assignedr(S1,_,R1,K),
    assignedr(S2,_,R2,K), R1 != R2.
:- same_rooms(S,S2), nrRoomMax(S,_), not nrRoomMax(S2,_).
:- assign_rooms(S1,R1), assignedr(S1,_,R2,_), R1 != R2.
:- assign_teachers(S1,T1), assignedt(S1,_,T2,_), T1 != T2.
:- serviceTeacher(T,P,N), #count { S,T
    :assignedt(S,_,T,_),part_sessions(P,S)} != N.
```

Listing 10.2: ASP model

C.3 – MIP model

$$\forall p \in P, R' = d_p^{P,R},\quad \forall s \in d_p^{P,S},\ \forall r \in R \setminus R',\ x_{s,r}^{S,R} = 0 \tag{1}$$

$$\forall p \in P, T' = d_p^{P,T},\quad \forall s \in d_p^{P,S},\ \forall t \in T \setminus T',\ x_{s,t}^{S,T} = 0 \tag{2}$$

$$\forall p \in P,\ \forall s \in d_p^{P,S}\quad min\_rooms_p^P \le \sum_{\forall r \in R} x_{s,r}^{S,R} \le max\_rooms_p^P \tag{3}$$

$$\forall p \in P,\ \forall s \in d_p^{P,S}\quad min\_lecturer_p^P \le \sum_{\forall t \in T} x_{s,t}^{S,T} \le max\_lecturer_p^P \tag{4}$$

$$\forall p \in P,\ \forall k \in K\quad size_k^K \le \sum_{r \in R} x_{s,r}^{S,R} * capacity_r^R \tag{5}$$

$$\forall s \in S:\quad x_s^{S,M} + (x_s^{S,D} - 1) * |M| + (x_s^{S,W} - 1) * |D| * |M| = x_s^{S,H} \tag{6}$$

$$forbidden\_period(S', H') = \forall s \in S',\ h = min(H'),\quad s_h \in \{0,1\}\quad (x_s^{S,H} - h) \ge ((length_s^S + M) * s_h - M)\quad (h - x_s^{S,H}) \ge ((|H'| + M) * s_h - M) \tag{7}$$

$$forbidden\_rooms(S', R') = \forall s \in S'\quad \forall r \in R',\ x_{s,r}^{S,R} = 0 \tag{8}$$

$$same\_weekday(S') = \forall s_1, s_2 \in S', s_1 < s_2\quad x_{s_1}^{S,D} - x_{s_2}^{S,D} = 0 \tag{9}$$

$$same\_start(S') = \forall s_1, s_2 \in S', s_1 < s_2\quad (x_{s_1}^{S,H} - x_{s_2}^{S,H}) = 0 \tag{10}$$

$$same\_rooms(S') = \forall s_1, s_2 \in S', s_1 < s_2\quad x_{s_1}^{S,R} = x_{s_2}^{S,R} \tag{11}$$

$$assign\_rooms(S', R') = \forall s \in S',\quad \forall r \in R, x_{s,r}^{S,R} = 1 \tag{12}$$

$$different\_rooms(S') = \forall s_1, s_2 \in S',\quad \forall r \in R,\ x_{s_1,r}^{S,R} + x_{s_2,r}^{S,R} \le 1 \tag{13}$$

$$periodic(S') = \forall s_1, s_2 \in S', s_1 < s_2,\quad (x_{s_1}^{S,H} + n) = x_{s_2}^{S,H} \tag{14}$$

$$sequenced(S_1, S_2) = \forall s_1 \in S_1,\ s_2 \in S_2\quad (x_{s_1}^{S,H} + length_{s_1}^S) \le x_{s_2}^{S,H} \tag{15}$$

$$required\_teachers(S', LG) = \forall s S',\ \forall t \in T,\ \forall \sigma \in \mathbb{N}\quad (\textstyle\sum_{s \in P} x_{s,t}^{S,T}) \le \sigma \tag{16}$$

$$\begin{aligned} & no\_overlap\_room(S') = \forall s_1, s_2 \in S', \\ & \quad ord_{s_1,s_2}, ord_{s_2,s_1} \in \{0,1\}, r_1 r_2 \in \{0,1\} \\ & \quad r_1 r_2 \le x_{s_1}^{S,R}, r_1 r_2 \le x_{s_2}^{S,R},\ r_1 r_2 \ge (x_{s_1}^{S,R} + x_{s_2}^{S,R} - 1) \\ & \quad r_1 r_2 \ge ord_{s_1,s_2}, r_1 r_2 \ge ord_{s_2,s_1} \\ & \quad (x_{s_1}^{S,H} - x_{s_2}^{S,H}) \ge ((length_{s_2}^S + M) * ord_{s_1,s_2} - M) \\ & \quad (x_{s_2}^{S,H} - x_{s_1}^{S,H}) \ge ((length_{s_1}^S + M) * ord_{s_2,s_1} - M) \end{aligned} \tag{17}$$

$$\begin{aligned} & no\_overlap\_group(S') = \forall s_1, s_2 \in S', \\ & \quad ord_{s_1,s_2}, ord_{s_2,s_1} \in \{0,1\}, ord_{s_1,s_2} + ord_{s_2,s_1} = 1 \\ & \quad (x_{s_1}^{S,H} - x_{s_2}^{S,H}) \ge ((length_{s_2}^S + M) * ord_{s_1,s_2} - M) \\ & \quad (x_{s_2}^{S,H} - x_{s_1}^{S,H}) \ge ((length_{s_1}^S + M) * ord_{s_2,s_1} - M) \end{aligned} \tag{18}$$

Appendix D – Complete list of instances

name	$\lvert R\rvert$	$\lvert T\rvert$	$\lvert U\rvert$	$\lvert S\rvert$	#ru	CP		ASP		CASP		MIP	
						BT	ST	BT	ST	BT	ST	BT	ST
gi5167	87	4	33	81	19	0.86	0.97	55.41	1.93	0.34	0.06	0.04	0.11
gi678	60	101	30	117	22	0.86	0.89	48.04	2.61	0.60	0.10	0.13	0.21
gi8201	82	4	33	118	23	0.86	1.86	15.71	0.08	0.30	0.23	0.17	0.26
gi8445	83	12	45	110	25	0.93	2.44	54.78	1.39	0.62	0.05	0.21	0.33
gi5301	99	96	55	139	33	0.85	483.59	68.70	0.36	0.40	0.12	0.22	0.49
gi9456	102	15	73	186	27	1.02	2.74	190.18	15.01	2.35	0.28	0.29	0.52
gi42910	84	96	110	163	30	1.03	1.87	146.45	1.25	0.49	0.18	0.16	0.30
gi29910	100	33	49	163	33	1.03	11.14	129.40	4.17	1.10	0.14	0.43	0.83
gi8601	104	97	143	200	16	0.99	0.98	42.49	4.93	0.40	0.17	0.08	0.14
gi6012	125	97	65	175	36	1.06	2.66	296.48	11.96	1.38	0.66	0.25	0.94
gi6101	85	100	48	212	40	0.99	1.47	131.63	8.63	0.78	0.38	0.18	0.36
gi4867	96	21	62	211	30	0.94	2.10	214.09	7.74	2.38	0.23	0.97	1.91
gi1045	108	99	50	175	44	0.93	7.60	84.57	2.49	0.98	0.20	0.26	0.44
gi2123	128	98	61	240	29	1.03	1.87	333.32	36.93	6.69	0.37	1.09	1.37
gi8056	104	4	66	223	37	1.06	5.01	221.33	7.81	3.31	0.57	1.61	2.52
gi5534	87	22	194	189	24	1.07	3.90	71.02	2.00	0.87	0.12	0.73	1.72
gi2501	123	98	151	245	23	1.00	1.50	143.26	28.74	1.53	0.49	0.22	0.38
gi223	102	6	135	219	30	1.01	4.09	87.73	1.52	0.89	0.19	1.88	2.79
gi1378	95	37	68	244	46	1.15	3.98	227.07	4.96	2.03	0.43	3.21	4.64
gi8467	115	24	67	264	46	1.30	3.33	297.07	10.62	4.23	0.57	2.48	3.87
gi7967	114	101	206	192	15	1.16	4.37	406.36	24.56	3.42	0.12	0.78	1.44
gi9123	78	4	66	325	38	1.15	213.82	212.85	17.56	1.51	0.78	1.81	2.81
gi9434	99	4	65	334	38	0.93	1.73	208.71	29.71	1.77	1.22	0.59	1.30
gi2245	110	105	132	313	30	1.08	2.25	134.36	16.68	2.01	0.26	1.16	1.94
gi8478	90	96	138	295	27	1.09	3.63	475.86	19.95	4.69	0.23	4.18	6.00
gi9334	107	14	86	341	30	1.03	2.24	162.71	30.17	4.54	1.43	1.63	2.64
gi745	107	105	91	352	38	1.10	1.95	60.29	7.10	0.99	3.89	0.34	0.54
gi1245	132	5	152	275	30	1.19	4.44	287.89	13.60	3.30	0.81	3.93	5.81
gi27910	97	100	114	369	34	1.07	59.67	131.10	15.43	1.53	1.01	0.98	1.74
gi5501	83	17	123	361	31	1.03	2.20	71.03	3.26	0.81	1.03	1.39	2.62
gi1867	86	39	91	405	47	1.08	5.22	77.95	12.20	1.99	4.64	2.07	2.95
gi4067	84	95	154	363	21	1.17	3.25	118.31	12.65	1.01	0.94	0.39	0.70
gi5856	78	10	114	324	50	1.12	5.15	182.23	6.41	2.02	0.40	5.19	8.20
gi9323	85	8	194	335	39	1.25	3.81	331.39	11.30	3.40	0.44	6.08	9.02
gi8956	102	97	162	419	26	1.25	11.72	527.65	33.91	1.26	1.99	1.79	2.84
gi8023	93	30	150	392	38	1.09	5.48	116.28	15.60	0.84	1.54	0.56	1.20
gi6889	94	98	66	522	55	1.24	3.64			4.33	10.41	5.76	6.45
gi3167	89	95	194	422	27	1.14	3.42	283.10	32.00	3.94	1.41	2.93	3.85
gi878	87	98	97	437	41	1.18	3.01	346.90	72.95	4.74	1.75	2.05	2.69
gi9189	82	27	319	362	52	1.36	7.84	395.76	14.18	3.58	3.39	6.88	10.05
gi1578	127	99	195	461	31	1.14	2.35	319.65	90.44	2.56	2.16	0.80	1.44
gi8678	83	24	261	369	28	1.24	8.08	525.14	38.13	3.69	4.99	10.32	15.92
gi7445	104	7	196	423	58	1.24	5.02	599.88	41.23	12.32	1.21	19.15	28.57
gi3934	104	96	159	623	64	1.32	4.56	613.68	142.05	5.51	2.02	5.84	8.20
gi5378	88	21	158	665	59	1.32	4.68	726.97	37.24	2.96	3.19	3.73	6.07
gi6745	98	98	150	624	65	1.35	7.20	634.56	125.36	13.92	4.53	10.29	14.84
gi4389	65	6	174	662	33	1.47	4.62	693.40	210.58	14.91	4.93	7.66	10.23
gi6278	99	8	192	588	40	1.22	17.32	491.72	183.93	13.94	3.68	6.37	9.01
gi7923	96	96	155	590	45	1.32	6.38	528.75	76.06	3.51	2.98	5.72	8.36
gi7867	118	101	211	617	59	1.51	8.89	1058.61	100.78	22.35	1.43	15.72	22.05
gi30910	104	9	190	523	49	1.22	8.45	472.29	22.97	7.81	1.21	21.71	32.06
gi4323	120	23	170	629	39	1.40	14.42	498.26	117.77	15.58	2.82	11.04	16.85
gi8745	108	105	249	632	34	1.38	8.38	610.86	156.25	20.13	3.22	13.08	20.08
gi2401	122	104	152	696	29	1.41	4.97			6.31	4.81	5.45	7.72
gi2867	102	11	154	783	38	1.46	4.74			6.41	11.81	6.53	10.06
gi4245	67	46	187	707	53	1.34	8.32	386.84	23.77	3.00	3.73	12.49	17.63
gi6134	113	105	152	748	35	1.40	8.56	437.44	109.94	3.17	13.31	4.95	7.36
gi8145	96	100	268	477	42	1.21	36.84	196.38	8.18	3.59	2.05	11.62	18.11
gi4278	95	47	251	811	20	1.30	4.93	645.57	25.87	1.75	8.60	1.08	2.09
gi7156	125	98	486	711	23	1.41	9.81	491.04	206.26	19.09	2.32	13.48	14.97
gi5145	99	96	154	1017	101	1.43	6.75			2.86	36.03	0.07	0.14

(. . .)

name	$\|R\|$	$\|T\|$	$\|U\|$	$\|S\|$	#ru	CP		ASP		CASP		MIP	
						BT	ST	BT	ST	BT	ST	BT	ST
(continued)													
gi1067	113	56	376	716	56	1.39	9.23	255.67	22.65	7.37	4.12	22.45	33.70
gi3367	98	45	539	836	99	1.48	8.44	372.95	16.04	5.09	6.10	22.19	94.02
gi30910	104	95	361	810	78	1.62	13.50	825.29	36.13	6.77	5.40	24.78	29.16
gi1778	112	10	400	880	109	1.66	11.05			46.16	15.64	65.96	107.49
gi7267	89	98	298	1295	30	1.67	10.51			33.20	29.72	35.02	258.28
gi77910	103	97	422	2120	102	2.17	12.58			114.51	94.53	96.28	141.20
gi9912	84	103	562	1956	172	2.08	47.84			10.48	1426.66		
gi2767	92	21	501	2003	78	2.02	30.48			128.21	57.11	130.51	7995.32
gi7034	102	100	454	2268	162	2.07	20.72			10.70	104.37	30.92	52.49
gi9023	144	103	1247	1980	200	2.49	24.85			142.35	41.64	258.11	799.06
gi7723	89	55	432	2745	152	2.50	28.02			79.88	218.58	157.51	405.68
gi5567	94	94	1770	6180	562	4.75	381.78			122.89	3548.58		
gi467	90	61	1982	8886	383	4.61	33.59			92.65	2609.97		
gi2934	106	73	2111	9168	515	5.34	33.87			92.25	2469.69		

Table 12: List of instances.

References

1. Scheduling zoo, http://schedulingzoo.lip6.fr/
2. Dataset Generator ITC-2007 (2022), https://cdlab-artis.dbai.tuwien.ac.at/papers/cb-ctt/
3. RobinX three field (2022), https://robinxval.ugent.be/RobinX/threeField.php
4. UTP dataset (2024), https://ua-usp.github.io/timetabling/instances
5. Abdullah, S., Turabieh, H.: On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems. Information Sciences **191**, 146–168 (2012). https://doi.org/https://doi.org/10.1016/j.ins.2011.12.018, https://www.sciencedirect.com/science/article/pii/S0020025511006670, data Mining for Software Trustworthiness
6. Amintoosi, M., Haddadnia, J.: Feature selection in a fuzzy student sectioning algorithm. In: 4th International Conference on the Practice and Theory of Automated (PATAT) 2004. pp. 147–160 (08 2004). https://doi.org/10.1007/11593577_9
7. Banbara, M., Inoue, K., Kaufmann, B., Okimoto, T., Schaub, T., Soh, T., Tamura, N., Wanko, P.: teaspoon: solving the curriculum-based course timetabling problems with answer set programming. Ann. Oper. Res. **275**(1), 3–37 (2019). https://doi.org/10.1007/s10479-018-2757-7
8. Baral, C.: Knowledge representation, reasoning and declarative problem solving. Cambridge university press (2003)
9. Barichard, V., Behuet, C., Genest, D., Legeay, M., Lesaint, D.: A Constraint Language For University Timetabling Problems. In: 13th International Conference on the Practice and Theory of Automated (PATAT) 2022. Louvain, Belgium (Aug 2022), https://hal.science/hal-04073981
10. Battistutta, M., Ceschia, S., De Cesco, F., Di Gaspero, L., Schaerf, A., Topan, E.: Local search and constraint programming for a real-world examination timetabling problem. In: Hebrard, E., Musliu, N. (eds.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 69–81. Springer International Publishing, Cham (2020)
11. Bellio, R., Ceschia, S., Di Gaspero, L., Schaerf, A.: Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling. Computers & Operations Research **132**, 105300 (2021). https://doi.org/https://doi.org/10.1016/j.cor.2021.105300, https://www.sciencedirect.com/science/article/pii/S0305054821000927
12. Bettinelli, A., Cacchiani, V., Roberti, R., toth, P.: An overview of curriculum-based course timetabling. TOP **23**, 313–349 (2015). https://doi.org/https://doi.org/10.1007/s11750-015-0366-z

13. Cambazard, H., Hebrard, E., O'Sullivan, B., Papadopoulos, A.: Local search and constraint programming for the post enrolment-based course timetabling problem. Ann. Oper. Res. **194**(1), 111–135 (2012). https://doi.org/10.1007/s10479-010-0737-7
14. Carter, M.W., Laporte, G., Lee, S.Y.: Examination timetabling: Algorithmic strategies and applications. The Journal of the Operational Research Society **47**(3), 373–383 (1996), http://www.jstor.org/stable/3010580
15. Caselli, G., Delorme, M., Iori, M.: Integer linear programming for the tutor allocation problem: A practical case in a british university. Expert Systems with Applications **187**, 115967 (2022). https://doi.org/https://doi.org/10.1016/j.eswa.2021.115967, https://www.sciencedirect.com/science/article/pii/S095741742101318X
16. Castro, C., Manzano, S.: Variable and Value Ordering When Solving Balanced Academic Curriculum Problems. ARXIV (Nov 2001)
17. Chen, M., Sze, S., Goh, S.L., Sabar, N., Kendall, G.: A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities. IEEE Access **PP**, 1–1 (Jul 2021). https://doi.org/10.1109/ACCESS.2021.3100613
18. Chiarandini, M., Di Gaspero, L., Gualandi, S., Schaerf, A.: The balanced academic curriculum problem revisited. Journal of Heuristics **18**(1), 119–148 (Feb 2012). https://doi.org/10.1007/s10732-011-9158-2, https://doi.org/10.1007/s10732-011-9158-2
19. Czarnecki, K., Østerbye, K., Völter, M.: Generative programming. In: Núñez, J.H., Moreira, A.M.D. (eds.) Object-Oriented Technology, ECOOP 2002 Workshops and Posters, Málaga, Spain, June 10-14, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2548, pp. 15–29. Springer (2002). https://doi.org/10.1007/3-540-36208-8_2, https://doi.org/10.1007/3-540-36208-8_2
20. David, S.: Optimal student sectioning on mandatory courses with various sections numbers. Annals of operations research **275**(1), 209–221 (2019)
21. Demirovic, E., Musliu, N.: Modeling high school timetabling as partial weighted maxsat. In: LaSh 2014: The 4th Workshop on Logic and Search (a SAT/ICLP workshop at FLoC 2014), July 18, Vienna, Austria (2014)
22. Demirovic, E., Stuckey, P.J.: Constraint programming for high school timetabling: A scheduling-based model with hot starts. In: van Hoeve, W.J. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26-29, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10848, pp. 135–152. Springer (2018). https://doi.org/10.1007/978-3-319-93031-2_10, https://doi.org/10.1007/978-3-319-93031-2_10
23. Demirović, E., Musliu, N.: Maxsat-based large neighborhood search for high school timetabling. Computers & Operations Research **78** (02 2017). https://doi.org/10.1016/j.cor.2016.08.004
24. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-shot asp solving with clingo. Theory and Practice of Logic Programming **19**(1), 27–82 (2019). https://doi.org/10.1017/S1471068418000054
25. Gogos, C., Dimitsas, A., Nastos, V., Valouxis, C.: Some insights about the uncapacitated examination timetabling problem. In: 2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM). pp. 1–7 (2021). https://doi.org/10.1109/SEEDA-CECNSM53056.2021.9566258
26. Goh, S.L., Kendall, G., Sabar, N.R.: Improved local search approaches to solve the post enrolment course timetabling problem. European Journal of Operational Research **261**(1), 17–29 (2017). https://doi.org/https://doi.org/10.1016/j.ejor.2017.01.040, https://www.sciencedirect.com/science/article/pii/S0377221717300759
27. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Annals of discrete mathematics, vol. 5, pp. 287–326. Elsevier (1979). https://doi.org/10.1016/S0167-5060(08)70356-X

28. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023), https://www.gurobi.com
29. ITC19: International Timetabling Competition (2019), https://www.itc2019.org/
30. Jawa Bendi, K., Junaidi, H.: simulated annealing approach for university timetable problem. Jurnal Ilmiah Matrik **21** (12 2019). https://doi.org/10.33557/jurnalmatrik.v21i3.723
31. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A.: Feature-oriented domain analysis (foda) feasibility study (01 1990)
32. Kiefer, A., Hartl, R.F., Schnell, A.: Adaptive large neighborhood search for the curriculum-based course timetabling problem. Annals of Operations Research **252**, 255 – 282 (2016), https://api.semanticscholar.org/CorpusID:20405903
33. Kingston, J.H.: Repairing high school timetables with polymorphic ejection chains. Annals of Operations Research **239**, 119–134 (2016)
34. Koné, O., Artigues, C., Lopez, P., Mongeau, M.: Event-based milp models for resource-constrained project scheduling problems. Computers & Operations Research **38**(1), 3–13 (2011)
35. Kristiansen, S., Sørensen, M., Stidsen, T.: Integer programming for the generalized high school timetabling problem. Journal of Scheduling **18** (08 2015). https://doi.org/10.1007/s10951-014-0405-x
36. Lemos, A., Monteiro, P., Lynce, I.: Disruptions in timetables: A case study at universidade de lisboa. Journal of Scheduling (02 2021). https://doi.org/10.1007/s10951-020-00666-3
37. Lewis, R., Paechter, B., Mccollum, B.: Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. Cardiff University, Cardiff Business School, Accounting and Finance Section, Cardiff Accounting and Finance Working Papers (01 2007)
38. Lindahl, M., Stidsen, T., Sørensen, M.: Quality recovering of university timetables. European Journal of Operational Research **276**(2), 422 – 435 (2019). https://doi.org/https://doi.org/10.1016/j.ejor.2019.01.026, http://www.sciencedirect.com/science/article/pii/S0377221719300451
39. Lü, Z., Hao, J.K.: Adaptive tabu search for course timetabling. European Journal of Operational Research **200**, 235–244 (01 2010). https://doi.org/10.1016/j.ejor.2008.12.007
40. Mccollum, B., McMullan, P., Paechter, B., Lewis, R., Schaerf, A., Di Gaspero, L., Parkes, A., Qu, R., Burke, E.: Setting the research agenda in automated timetabling: The second international timetabling competition. INFORMS Journal on Computing **22**, 120–130 (02 2010). https://doi.org/10.1287/ijoc.1090.0320
41. Müller, T., Rudová, H., Müllerová, Z.: University course timetabling and International Timetabling Competition 2019. In: Burke, E.K., Di Gaspero, L., McCollum, B., Musliu, N., Özcan, E. (eds.) Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2018). pp. 5–31 (2018)
42. Müller, T., Murray, K.: Comprehensive approach to student sectioning. Annals of Operations Research **181**, 249–269 (Dec 2010). https://doi.org/10.1007/s10479-010-0735-9
43. Nagata, Y.: Random partial neighborhood search for the post-enrollment course timetabling problem. Computers & Operations Research **90**, 84–96 (2018). https://doi.org/https://doi.org/10.1016/j.cor.2017.09.014, https://www.sciencedirect.com/science/article/pii/S0305054817302447
44. Nešić, D., Krüger, J., Stănciulescu, u., Berger, T.: Principles of feature modeling. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. p. 62–73. ESEC/FSE 2019, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3338906.3338974, https://doi.org/10.1145/3338906.3338974
45. Nysret Musliu, E.D.: Solving high school timetabling with satisability modulo theories (08 2014), https://api.semanticscholar.org/CorpusID:14768396

46. Ostrowski, M.: Modern constraint answer set solving. Ph.D. thesis, Universität Potsdam (2018)
47. Árton P. Dorneles, de Araújo, O.C., Buriol, L.S.: A fix-and-optimize heuristic for the high school timetabling problem. Computers & Operations Research **52**, 29–38 (2014). https://doi.org/https://doi.org/10.1016/j.cor.2014.06.023, https://www.sciencedirect.com/science/article/pii/S0305054814001816
48. Post, G., Ahmadi, S., Daskalaki, S., Kingston, J., Kyngas, J., Nurmi, K., Post, G., Ahmadi, S., Daskalaki, S., Kyngas, J., Ranson, D.: An XML format for benchmarks in High School Timetabling. Annals of Operations Research **194**, 385–397 (Apr 2012). https://doi.org/10.1007/s10479-010-0699-9
49. Prud'homme, C., Fages, J.G.: Choco-solver: A java library for constraint programming. Journal of Open Source Software **7**(78), 4708 (2022). https://doi.org/10.21105/joss.04708, https://doi.org/10.21105/joss.04708
50. Rubio, J.M., Palma, W., Rodriguez, N., Soto, R., Crawford, B., Paredes, F., Cabrera, G.: Solving the Balanced Academic Curriculum Problem Using the ACO Metaheuristic. Mathematical Problems in Engineering **2013**, e793671 (Dec 2013). https://doi.org/10.1155/2013/793671, https://www.hindawi.com/journals/mpe/2013/793671/, publisher: Hindawi
51. Say Leng Goh, G.K., Sabar, N.R.: Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem. Journal of the Operational Research Society **70**(6), 873–888 (2019). https://doi.org/10.1080/01605682.2018.1468862, https://doi.org/10.1080/01605682.2018.1468862
52. Schindl, D.: Optimal student sectioning on mandatory courses with various sections numbers. Annals of Operations Research **275** (04 2019). https://doi.org/10.1007/s10479-017-2621-1