Using model-driven techniques for specifying and solving optimisation problems in the healthcare domain: A case study based on the Integrated Healthcare Timetabling Problem

Maximilian Kratz
Real-Time System Lab, Technical University of Darmstadt
Darmstadt, Germany
maximilian.kratz@es.tu-darmstadt.de

Jule Pfau
Technical University of Darmstadt
Darmstadt, Germany
jule.pfau@stud.tu-darmstadt.de

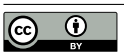
Steffen Zschaler
Department of Informatics, King's College London
London, United Kingdom
szschaler@acm.org

Jens Kosiol
Chair of Practical computer science/software systems engineering, Brandenburg University of Technology Cottbus – Senftenberg
Cottbus, Germany
jens.kosiol@b-tu.de

Andy Schürr
Real-Time System Lab, Technical University of Darmstadt
Darmstadt, Germany
andy.schuerr@es.tu-darmstadt.de

Abstract

Optimisation problems are ubiquitous in healthcare, underlying decision-making in a variety of contexts like resource allocation, scheduling, and treatment planning. However, there is sufficient variation between individual healthcare contexts and their specific associated optimisation problems to make reuse of solutions, and even problem specifications, difficult. Moreover, optimisation problems are typically specified in mathematical terms or in programming code, which is often inaccessible to clinical experts. Model-driven techniques have been used in other fields of engineering to tackle similar problems by providing means to express specifications at an appropriate level of abstraction and in terms suitable for the modelled domain. Therefore, it is promising to explore how far Model-Driven Optimisation can help in the healthcare domain. We do this in the context of the Integrated Healthcare Timetabling Problem, a challenging resource-allocation problem in the healthcare domain presented at the Integrated Healthcare Timetabling Competition 2024. Our solution, based on the Graph-Based (M)ILP Problem Specification tool, uses class diagrams to model domain concepts, the Object Constraint Language to define domain constraints, and Graph Transformation rules to capture available decisions. Such a specification can be translated into a Mixed Integer Linear Programming formulation that can be solved by the off-the-shelf solver Gurobi. Our solution solves all 30 benchmark instances within the original time limit of 10 min, and two of the public instances to optimality with a relaxed time limit, but is not yet as efficient as hand-crafted problem formulations. It also provides a more accessible and flexible problem specification, which we believe can be further improved and automated in future work.



MODELS Companion 2026, Málaga, Spain

ACM ISBN 979-8-4007-2903-4/2026/10
https://doi.org/10.1145/3837062.3839356

CCS Concepts

• **Applied computing → Health informatics**; • **Software and its engineering → Domain specific languages**; • **Mathematics of computing → Combinatorial optimization**; *Graph theory*; • **Computing methodologies → Optimization algorithms**.

Keywords

Integrated Healthcare Timetabling Competition (IHTC) 2024, Model-Driven Optimisation, Optimisation Problem Specification, Mixed-Integer Linear Programming, Graph-Based (M)ILP Problem Specification Tool, Optimisation in Healthcare

ACM Reference Format:
Maximilian Kratz, Jule Pfau, Steffen Zschaler, Jens Kosiol, and Andy Schürr. 2026. Using model-driven techniques for specifying and solving optimisation problems in the healthcare domain: A case study based on the Integrated Healthcare Timetabling Problem. In *ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion 2026), October 04–09, 2026, Málaga, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3837062.3839356

1 Introduction

Optimisation problems are ubiquitous in the healthcare domain [31], ranging across such diverse areas as allocation and scheduling problems [1, 30], optimising overall procedures and workflows [3], or the optimal spatial distribution of healthcare providers [20]. Good solutions can have crucial impact on the quality and cost of provided services. Yet, specifying and solving optimisation problems is hard. It requires combining a very good understanding of healthcare systems with expert knowledge about mathematical modelling and optimisation. A formal specification of the problem is required that (i) makes the problem *tractable* for a solver by using mathematical abstractions and applying appropriate solving strategies; and (ii) makes the implementation *usable* by developing pipelines integrating into an existing environment, for reading out the data required and making computed solutions accessible. Context is

essential for quality improvement in healthcare [16], meaning that optimisation-problem specifications need to be adapted for different wards, hospitals, countries, etc. This is extremely costly with current approaches, which contributes to a lack of adoption of mathematical optimisation approaches in clinical practice [7].

Model-driven (software) engineering has been developed to raise the abstraction level in engineering efforts, enabling users to express the system to be developed in terms of concepts of the respective domain [10]. Research on applying model-driven approaches and techniques for specifying and solving optimisation problems—sometimes referred to as Model-Driven Optimisation (MDO)—has emerged over the last decade [8, 9, 21, 23, 38] and, among other things, this approach raises hope to make context-specific adaptations easier [26]. However, transfer of these ideas into the healthcare domain remains limited. We are interested, in particular, in understanding if and how a model-driven approach can enable experts from the healthcare domain to co-create and adapt specifications and solutions to optimisation problems. In this paper, we start by using the Integrated Healthcare Timetabling Problem (IHTP)—formulated as part of the Integrated Healthcare Timetabling Competition (IHTC) 2024 [14]—as a case study. It combines several allocation problems typical for the healthcare domain, namely allocating (i) patients to available surgery slots, (ii) surgeons to surgery slots with allocated patients, (iii) patients to hospital rooms, and (iv) nurses to hospital rooms.

Concretely, we ask the following research questions:

RQ1 Can established declarative modelling formalisms—namely conceptual modelling using class diagrams, the Object Constraint Language (OCL), and declarative model transformations—be used to express complex optimisation problems from the healthcare domain (in terms close to the domain)?

RQ2 Can model-based, declarative specifications be transferred to specifications in established, off-the-shelf solvers like Gurobi?

RQ3 Given a model-based specification and such a translation to a solver, can solutions be efficiently computed?

After clarifying some background in Section 2, we address **RQ1** in Section 3 and find that we can compactly and declaratively specify the IHTP using class diagrams, Graph Transformations (GTs), and only a small extension to the syntax of OCL. Importantly, this specification directly uses domain terminology. MDO research has already established that it is relatively straightforward to use evolutionary algorithms (and similar meta-heuristic approaches) to solve model-based optimisation problems [8, 11, 23]. In contrast, in Section 4 (and to answer **RQ2**), we apply the Graph-Based (M)ILP Problem Specification (GIPS) tool [18] to translate the specification into a Mixed Integer Linear Programming (MILP) problem and use Gurobi[1] to compute (near) optimal solutions. While some of this translation currently remains manual, we find promising potential for automation. We then compute solutions to the IHTP, largely following the contest rules for the IHTC 2024. We find that, while our approach is able to compute a valid solution for each of the contest instances, our runtimes are not yet competitive. To close our paper, we discuss related work in Section 6 and summarise our findings and directions for future research in Section 7. The work presented in this paper is also available as an extended preprint [25].

[1]Gurobi Optimizer: https://www.gurobi.com/solutions/gurobi-optimizer (Accessed June 17, 2026)

2 Background and Running Example

In general, specifying optimisation problems manually is a hard and error-prone task. It requires knowledge about the problem domain that should be modelled as well as the mathematical background of optimisation itself. To make this process accessible to domain experts such as hospital administrators, an approach is needed that allows optimisation tasks to be specified without extensive technical knowledge. This is what MDO addresses by combining Model-Driven Engineering (MDE) and optimisation [8, 23, 38]. An optimisation problem typically has three key components [27]: *Decision variables* model the problem aspects that can be controlled, such as the assignment of a patient to a specific room on a given day. *Constraints* restrict the values of variables combinatorially and, therefore, define the search space. A feasible solution, i.e., a solution that fulfils all (hard) constraints, can be ranked by an *objective value*. In MDO, we start with a metamodel—a conceptual model of the problem domain, technically in the form of a Unified Modeling Language (UML) class diagram. A constraint formalism—namely the Object Constraint Language (OCL) [12]—serves to express constraints and optimisation objectives. Instance models of the metamodel define concrete problem instances and solutions to these. *Model transformations*—describing potential manipulations of instance models—serve as decision variables, capturing selectable steps to transform a problem instance into a feasible solution. This chapter explains these fundamental concepts and technologies necessary for understanding our overall MDO approach to creating the key components presented in this paper. In Section 2.1 we present the running example and in Section 2.2, we briefly explain MILP. Finally, in Section 2.3, we concisely present our MDO framework of choice, GIPS, which integrates GT with MILP and which we used to implement our approach for this paper.

2.1 The Integrated Healthcare Timetabling Problem

The running example, which we use as a case study in this paper, is the IHTP. This problem domain proposed by the IHTC [14] integrates three $\mathcal{NP}$-hard subproblems, which are interdependent to varying degrees. Concretely, they involve a fixed time period during which one must determine (i) the admission of patients to fixed rooms, where mandatory patients must be scheduled, and optional ones can be postponed to the next period; (ii) a timely Operating Theatre (OT) assignment that determines the surgery date and location; and (iii) the allocation of nurses to all occupied rooms. The valid assignment of all required resources is governed by eight hard constraints, which define mandatory conditions that must be satisfied, for instance, ensuring that room capacities are not exceeded. Beyond that, the allocations should be optimised according to a set of eight soft constraints that define objectives, such as minimising the wait time of patients prior to their admission.

2.2 Mixed-Integer Linear Programming

Optimisation is a fundamental concept within Operations Research (OR) that deals with identifying the best possible solution to a problem from a set of feasible alternatives. This is formalised through a mathematical model comprising decision variables, an objective function, and a set of constraints that represent the problem's limitations or requirements. The objective function evaluates the quality of a solution, whereas constraints restrict the feasible region.

A widely used formalism for specifying and solving optimisation problems is MILP [28, 37]. In this formalism, both the objective function and the constraints of the problem have to be linear. The decision variables can be of two types: continuous variables describe divisible resources, and integer variables capture inherently discrete or binary decisions. This combination allows MILP to represent both resource allocation and combinatorial decision-making. Formally, a MILP can be written as

$$\begin{aligned} \max_{x\in\mathbb{Z}^p,\ y\in\mathbb{R}^q} \ & F(x,y) = c_x^\top x + c_y^\top y \\ \text{subject to} \quad & A_x x + A_y y \le b, \quad x, y \ge 0. \end{aligned} \tag{1}$$

In this notation, x denotes the integer-valued variables, encoding discrete or binary decisions, y denotes the continuous variables for divisible resources, c_x, c_y are vectors of objective coefficients, with c_j indicating the contribution of variable x_j or y_j to the objective function, A_x, A_y are submatrices of the constraint coefficient matrix that quantify the impact of integer and continuous variables on each constraint, and $b \in \mathbb{R}^m$ is the right-hand side vector specifying the available capacity or limit for each constraint. The condition $x, y \ge 0$ ensures non-negativity, which is essential if variables represent quantities such as production volumes or time allocations. The intersection of all linear constraints determines the set of all feasible solutions, and the goal of the optimisation is to find a solution that maximises or minimises the objective function. If a solution (x^*, y^*) exists such that no other feasible pair achieves a better objective value, it is called optimal.

State-of-the-art optimisation solvers such as *Gurobi* provide highly optimised implementations of both exact and heuristic methods. They combine a wide range of algorithmic techniques to efficiently explore the solution space and handle large-scale real-world MILP instances [22]. Each IHTP problem instance can be modelled as an MILP problem, with decision variables representing assignments such as patient-created workloads to nurse shifts, subject to linear constraints. This enables the use of modern solvers to compute feasible / optimal solutions for complex healthcare scheduling problems. Since there are many different IHTP problem instances to be solved, the need for an automatic translation program from a problem instance to a corresponding MILP formulation arises.

2.3 Graph-Based (M)ILP Problem Specification

The size of an MILP problem grows rapidly with increasing model complexity, making even relatively small instances difficult to manage by hand. As a result, state-of-the-art approaches rely on code generators that are tailored to specific problem domains (for example, [35]). Manually creating such domain-specific MILP problem generators, however, remains a challenging and costly task. In addition, it requires expertise in mathematical optimisation as well as in the specific problem domain. Therefore, there is great interest in tools that can simplify this complex process and make it accessible to users with little or no knowledge of MILP, allowing them to implement an MDO approach based on MILP.

With this goal in mind, we developed GIPS [17, 18, 24, 32]. Following an MDE approach, GIPS has its own Domain-Specific Language (DSL)—Graph-Based (M)ILP Problem Specification Language (GIPSL)—providing an accessible way to specify optimisation problems (and their related domain concepts) based on an underlying Eclipse Modeling Framework (EMF)-based metamodel. It uses graph transformations to perform modifications to a given input model by searching for GT rule matches—places in the model, where a rule can be applied—and using the MILP solver to decide at which of these GT rules should be applied to obtain the optimised model. A GIPSL specification broadly consists of GT rules to perform model-to-model transformations on the input model, *mappings* to assign MILP variables to rule matches and control their application, *constraints* that control the selected mapping variables, and *functions* to compute an objective value. Based on this high-level specification and an input model, GIPS automatically generates the corresponding executable code and the MILP formulation, which is then passed to a state-of-the-art MILP solver. This integration allows the automated formulation of complex optimisation problems implementing MDO, reducing the likelihood of modelling errors, and significantly decreasing the effort required to adapt the formulation when the problem definition changes [17, 26].

3 Modelling an Integrated Optimization Scenario

In this section, we address **RQ1** (Can established declarative modelling formalisms be used to express complex optimisation problems from the healthcare domain?) by presenting our specification of the IHTP problem, comprising three main components: The metamodel serves as the basis for defining the structural properties of the problem, while GT rules describe local valid modifications to an input model, and OCL constructs specify global requirements, as well as our optimisation metrics. We detail each of these elements in the following. The resulting specification is amenable to different approaches to computing solutions to concrete instances. Subsequently, in Section 4, we briefly show one specific approach based on MILP and the GIPS tool.

3.1 Metamodel

The definition of the IHTP is provided in natural-language text [14], describing the relevant groups of persons, the associated objects, and the constraints controlling their interaction. Here, we translate this into a precise and formal metamodel, which we show in Figure 1 using UML class diagram notation. The resulting complete metamodel is a representation of all necessary information of the IHTP in a structured form. Its classes comprise an extended set of attributes and references, which will be explained in the following paragraphs. All references displayed in Figure 1 as blue edges annotated with «derived» are to be identified through the optimisation process (and we number them by the subproblem they belong to). This means that a problem instance of the IHTP without any solution does not contain any of the «derived» edges shown

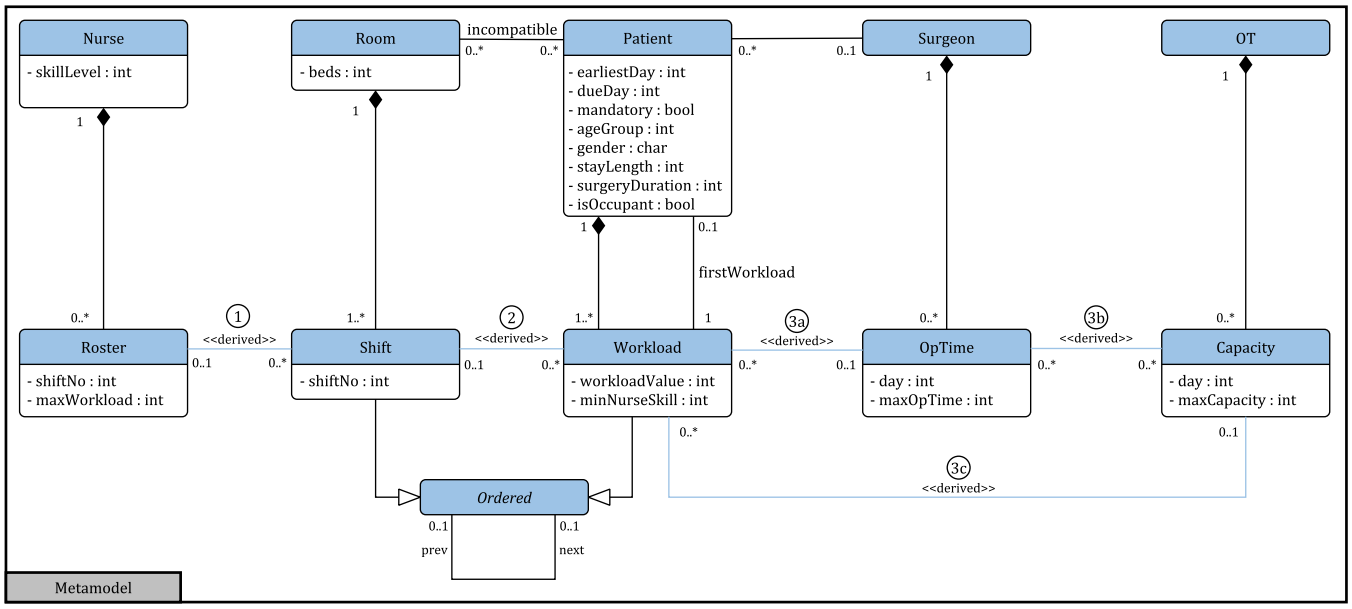


Figure 1: Complete initial (domain) metamodel for the IHTP.

in Figure 1. Most of the other (straightforward) references in the class diagram do not show their labels for improved clarity of the diagram. For example, the reference from the class `Workload` to the class `OpTime` ends with a multiplicity of `0..1` and can be addressed via the name `opTime`. The references with a multiplicity of `0..*` can be addressed with the plural of the target node's name, e.g., the `Capacity` nodes can be reached via the reference name `capacities` starting from the class `OpTime`. We introduce the metamodel by addressing one subproblem of the IHTP at a time.

For the modelling of our first subproblem, the *Nurse-to-Patient Assignment (NPA)*, we introduce a class `Nurse` that models a nurse with a specific `skillLevel` as an integer. A `Nurse` has a set of `Roster` objects that model the availability as well as the maximum workload of the nurse. For this, the class `Roster` includes an attribute `shiftNo` to indicate on which shift the nurse is available and the attribute `maxWorkload` to keep track of the maximum workload the nurse is capable of handling in this particular shift. On the other side of this sub-problem, there are the classes `Room` and `Shift`. An object of the class `Room` represents a single room of the hospital, whereas a `Shift` models one specific shift of a specific room. For a complete planning horizon of D days, there are $3D$ shift objects per `Room` in the complete model (3 shifts per day). For being able to represent the order of `Shift` objects, this class inherits from the abstract class `Ordered` that models a double-linked list via the references `prev` and `next`. The blue «`derived`» edge (1) between `Roster` and `Shift` models the assignment of `Roster` objects of a nurse to shifts of a specific room. As a result, it is possible to encode assignments of specific nurses' rosters to individual shifts of rooms.

For the second subproblem, the *Patient-to-Room Assignment (PRA)*, we add the classes `Patient` and `Workload`. A `Patient` object models an individual patient of the IHTP, including different attributes to model properties of the patients. `earliestDay` models the earliest possible admission date, whereas `dueDay` models the latest possible admission day, both representing the respective day number. The Boolean attribute `mandatory` is set to `true` for mandatory patients and `false` for optional patients. Age and gender are modelled similarly. Attribute `stayLength` keeps track of the length of the patient's stay measured in days. `surgeryDuration` models the surgery duration for the operation that is necessary for the respective patient. Lastly, the Boolean flag `isOccupant` indicates if the respective patient is an occupant and, therefore, had their surgery in the last planning horizon. Besides the mentioned attributes, the class `Patient` also contains a reference `incompatible` that can link to multiple objects of the class `Room`, with which a particular `Patient` is not compatible (e.g., due to missing equipment). Each `Patient` object contains a set of instances of class `Workload` that also inherit from the abstract super class `Ordered`. They model the workload created for nurses by the patient in each shift, each having a specific value (`workloadValue`) and also a minimum required nursing skill (`minNurseSkill`). The blue «`derived`» edge (2) between `Workload` and `Shift` models the assignment of the patient's workload (and, thus, the patient themself) to a specific room in an individual shift. The metamodel is, thus, capable of representing the assignment of a patient to multiple rooms for different shifts. For all occupants of a problem instance, the corresponding «`derived`» edges are predetermined by the input data.

The remaining classes model the last of the three subproblems, the *Surgical Case Planning (SCP)*. First, `Surgeon` represents individual surgeons working in the hospital. Each `Surgeon` contains a set of objects of type `OpTime`. `OpTime` captures the days on which the surgeon can operate (`day` attribute), and the maximum operation time available on each day (`maxOpTime` attribute). A `Patient` may be pre-allocated to a single `Surgeon`, and a `Surgeon` can be assigned to operate on multiple `Patients`. The assignment of the surgery of a specific `Patient` is modelled via the blue «`derived`» edge (3a) between `Workload` and `OpTime`. This reference connects

Listing 1: OCL constraint implementing the hard constraint H7 of the IHTP.

```
-- (H7): Room capacity
context Shift inv:
    shiftNo mod 3 = 0 implies
        workloads->size() <= room.beds
```

Listing 2: OCL constraint implementing the soft constraint S7 of the IHTP.

```
-- (S7): Admission delay
context Patient min:
    if firstWorkload.shift.oclIsUndefined() or
        isOccupant then 0
    else firstWorkload.shift.shiftNo / 3
        - earliestDay
```

the first workload of a patient with an available operation time slot of the surgeon. For allocating a surgery to an operating theatre, the metamodel provides a class OT that contains a set of Capacity objects, each modelling the availability of the OT on a specific day (using attributes day and maxCapacity). To model a specific assignment of a patient's surgery to an operating theatre, the remaining two blue «derived» edges (3b) and (3c) are necessary. The first edge (3b) between OpTime and Capacity describes the assignment of the corresponding surgeon to a specific operating theatre. The other edge (3c) between Workload and Capacity models the assignment of the patient to the respective operating theatre. Both assignments also set the specific day on which the surgery should take place.

The illustration in Figure 1 is slightly simplified because in the complete class diagram, there is an abstract superclass called NamedElement with a single String attribute used to assign names to objects. In fact, most of the classes like Nurse, Room, Patient, etc., inherit from the class NamedElement. We omitted this class in Figure 1 to reduce the size of the class diagram in this section.

The complete metamodel is also not able to enforce valid assignments of the *«derived»* edges, as these are subject to both local and global requirements. The two types of requirements can be defined through an additional technique able to enforce hard constraints on arbitrary model segments, for which we chose OCL.

3.2 OCL Specification

To ensure (global) model validity, all hard and soft constraints of the IHTP problem description can be expressed using OCL constraints. The comprehensive specification is based on the assumption that the provided model instances conform to the defined metamodel, enabling a compact definition that only considers valid assignments of *«derived»* edges.

As a first example, we take a look at the hard constraint H7 in Listing 1, which directly derives from the problem description. The shown invariant is only enforced for each morning shift, as indicated by the keyword *implies*. This is possible because patient admission and release are only allowed for the beginning of the first shift of a day as per the IHTP description. If the corresponding shift number is divisible by three (hence, the respective shift is a morning shift), the number of assigned workloads or, respectively, patients must satisfy the maximum room capacity.

Since we also need to specify the soft constraints of the IHTP, there is a need for a way to express (parts of) the objective function on the model level. The OCL standard does not define a way for specifying objective functions. Instead, we introduced an extension to OCL to be able to specify arithmetic expressions for a given context, which is inspired by Tomaszek [34]. Instead of specifying invariants in a specific context, the newly introduced keyword min (or max) can be used to state the direction of the optimisation process and it also defines that the following body belongs to a soft constraint. In contrast to the body of the hard OCL constraint, which contained a Boolean expression that must be satisfied at all times, the body of the soft constraint contains an arithmetic expression that evaluates to a numerical value. The idea behind this OCL extension is that for every instance of an object of the context in the model, the corresponding arithmetic expression will be evaluated. The sum of this evaluation will contribute to the overall objective function, which is the sum over all soft constraint definitions, controlled via the keyword min or max.

Listing 2 presents the soft constraint S7, which calculates the admission delay (to be minimised) for each patient. The calculated penalty is only valid if the patient was initially admitted, which is ensured if the first workload is associated with a shift. Furthermore, occupants are excluded as they are predetermined and scheduled from the start of the planning horizon, meaning they cannot be delayed. For the calculation of the penalised wait time, the body of the constraint first navigates from the Patient to their firstWorkload, afterwards the connected Shift, and lastly to the respective shift number (shiftNo). The value gets divided by 3 in order to get the respective day number. Note that this is an integer division, which truncates any fractional part of the result intentionally. Lastly, we subtract the earliest possible admission day of the Patient, which results in our penalty cost if we schedule the patient for a later day that is not the first possible admission day.

Like these examples, all further constraints of the IHTP problem description can be specified using OCL. This applies to the hard constraints as well as the soft constraints of the IHTP. The full specification of all OCL constraints can be found in [25].

With the specification of hard constraints and soft constraints via OCL, there is the requirement left to modify the model, i.e., to create «derived» edges. For this, we introduce GT rules.

3.3 Graph Transformation Rules

The proposed metamodel contains five *«derived»* edges, responsible for subtasks of the scheduling problem. We have defined a set of GT rules that can create each of these associations necessary for valid IHTP solutions. The complete set of rules is provided in [25], and Figure 2 as well as Figure 3 present the two GT rules defined for the PRA subproblem exemplarily.

The rule assignPatientToRoom_domain assigns the patient's first workload to a morning shift, as implied by the first attribute constraint. The second attribute constraint ensures that the admission day is the same as the day the surgery is scheduled. Furthermore, the Left-Hand Side (LHS) contains two Negative Application

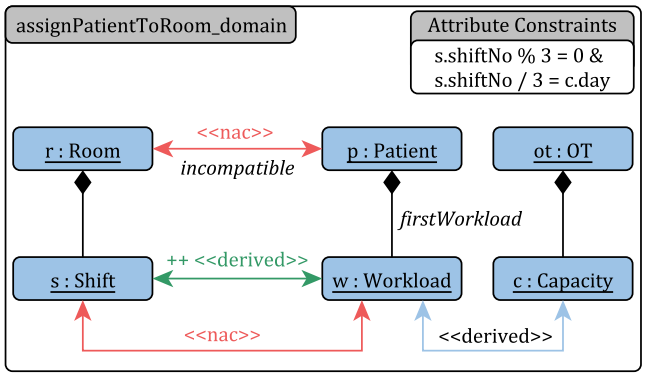


Figure 2: Domain GT rule for initial patient admission.

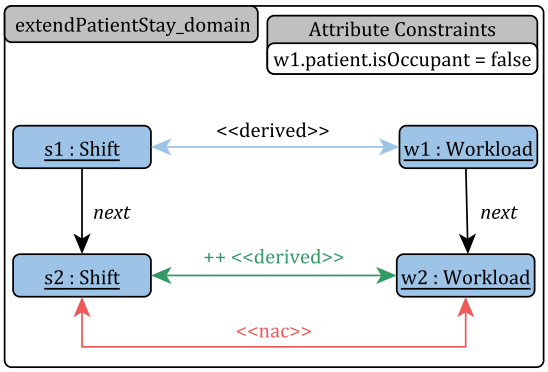


Figure 3: Domain GT rule for extending the patient's stay.

Conditions (NACs), where the first ensures only matches for compatible rooms are selected, and the second prevents patients from being admitted to the same `Shift` object multiple times. In summary, assigning the first workload determines the admission day and room for the duration of a patient's stay. Both attribute constraints were derived from some of the OCL constraints of the full specification. Subsequently, the rule `extendPatientStay_domain` assigns the patient's next workload to the consecutive shift associated with the same room, thereby extending their stay by one shift. The NAC once again prohibits multiple assignments between a shift and workload pair. As seen in Figure 3, to identify this next shift, the LHS relies on the *«derived»* edge created by the previous workload assignment. This always entails a consistent room assignment, as all shift objects in a list are associated with the same room. To assign all workloads of a patient, the second rule must be successively executed for every workload beyond the first. Consequently, each following match is directly dependent on the *«derived»* edge created by the prior application. To ensure termination of the complete rule set, all rules contain a NAC that forbids creating more than one `derived` edge between the same two nodes.

Similarly to the two exemplarily presented GT rules presented in this section, all necessary model transformations of all subproblems of the IHTP can be represented with GT rules.

4 Implementation with GIPS

The three proposed main components (metamodel, OCL constraints, and GT rules) comprehensively specify the IHTP and also define optimisation metrics through the OCL soft constraints. This specification poses a static representation of the IHTP. Importantly, it is *parametric* in the sense that the problem is defined generically: instances of the metamodel that do not contain *«derived»* references correspond to individual problem instances provided that the OCL constraints and GT rules uniformly define constraints, optimisation objectives, and decision variables for all these instances.

However, the specification itself does not incorporate a mechanism to execute the proposed GT rules and drive the optimisation. To this end, there are several different tools and approaches available. While technical details differ (i.e., translations to tool-specific formats and languages would be needed), the presented components principally suffice to compute solutions to IHTP instances in tools like MDEOptimiser [11], MOMoT [8], or VIATRA-DSE [21]. Notably, the just presented specification does not depend on a certain approach to solve optimisation problems, but various methods of state space exploration can be used. In particular, evolutionary algorithms and similar metaheuristic approaches can be applied rather directly, given our specification.

In this section, we address **RQ2** (Can model-based, declarative specifications be transferred to specifications in established, off-the-shelf solvers like Gurobi?) by briefly describing how the high-level, model-driven specification of IHTP, presented in Section 3 can be transformed into a GIPSL specification that can be solved automatically with the GIPS framework and an off-the-shelf MILP solver as part of a comprehensive pipeline (see Figure 4). To do this, we first have to translate the specification from Section 3 into a specification in GIPSL, the DSL used by GIPS. This is more than a change of representation: through this translation, we change from a problem formulation that is focused on the outcome (statically and declaratively describing *what* a solution should look like using constraints over the outcome model) to one that is focused on the process of finding that outcome (describing constraints over GT rule applications that govern *how* to optimally apply these rules).

A key challenge is that the original GT rules of Section 3 were designed for sequential execution, where later GT rule applications depend on structures created by earlier ones. Since an MILP solver selects all decisions simultaneously, these dependencies must be reified explicitly in the model. To achieve this, we introduce so-called *virtual nodes* within the metamodel that replace the original derived edges of the original metamodel of Section 3. Each virtual node represents a potential rule application and contains a Boolean attribute indicating whether it is selected as part of the final solution. Additional dependency relations between virtual nodes capture the execution order and prerequisite relationships that previously arose implicitly during sequential rule execution. This transformation allows all possible decisions to be represented statically before running the MILP-based optimisation procedure.

The next step is a so-called preprocessing phase, which can be seen on the left-hand side of Figure 4. Starting from a specific IHTP problem instance imported from a JSON file (which was transformed from the JSON file to a model with the help of the visualised *JSON2Model* transformation), GT rules are executed exhaustively

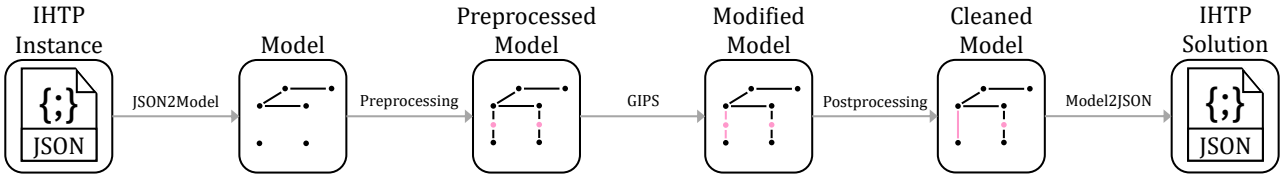


Figure 4: Pipeline illustrating the process from start to obtaining a final solution.

to generate all admissible virtual nodes and their dependencies. These preprocessing GT rules are systematically derived from the original domain rules. The resulting enriched model (see *Preprocessed Model* in Figure 4) contains a complete representation of all feasible assignment options together with the information needed to determine which combinations are valid. This shifts much of the search-space construction into a generic preprocessing step and avoids the need for manually coded, problem-specific MILP formulations.

After preprocessing, the enriched model is handled by the derived GIPSL specification. In the derivation of the GIPSL specification, the original OCL constraints that describe valid solution states are reformulated as constraints on the selection of virtual nodes based on the enriched model. Likewise, soft constraints are transformed into objective-function components in the GIPSL DSL. GIPS then maps possible GT rule applications to MILP decision variables and generates the corresponding optimisation model automatically. Although parts of the described translation were implemented manually in this work, we believe that the process is sufficiently systematic to be automated in the future.

The complete optimisation pipeline (see Figure 4) therefore consists of: (1) Importing the JSON data into a graph-based model (*JSON2Model*), (2) preprocessing the model to create virtual nodes including their inter-dependencies (*Preprocessing*), (3) generating an instance-specific MILP formulation from the generic IHTP GIPSL specification (*GIPS*), (4) solving the instance-specific MILP problem with Gurobi (*GIPS*), and (5) post-processing the selected virtual nodes into the final solution model (*Postprocessing*) as well as exporting the solution as JSON file (*Model2JSON*), if desired. The complete pipeline is implemented and runs automatically, whereby the step *GIPS* in Figure 4 is based on previous work on the GIPS tool [17, 24]. *Preprocessing* and *Postprocessing* are new contributions of this work. However, the current implementation of Pre- and Postprocessing is specific to the IHTP and, most importantly, we manually translated the metamodel and OCL from the previous section into GIPSL. By and large, however, our procedure in both implementing Pre- and Postprocessing and translating OCL to GIPSL were principled and followed identifiable patterns. Therefore, we are convinced that large parts of the pipeline can be made generic and that the translation to GIPSL can be automated (indeed, we have started implementing this automation).

Overall, this demonstrates how a domain-oriented specification based on metamodels, OCL constraints, and GT rules can be systematically transformed into a solver-executable MILP model given an instance model representing a specific problem to be solved. The approach preserves the high-level problem description while enabling (semi-)automated optimisation and is a foundation for a largely automated MDO workflow in future work. A more in-depth description including examples can be found in our preprint [25]. The complete GIPSL specification and some additional documentation can be found in our GitHub repository[2].

5 Optimizing IHTP Instances

As we said in the introduction, our focus is on the specification of the optimisation problem and how we can improve it. Nevertheless, solver performance is important for practical applications. Addressing **RQ3**, we report on some experimental results.

Throughout this experiment, we are adding increasingly complex soft constraints to the problem and, therefore, show the following selection of specifications that differ in the number of soft constraints they implement:

- *3SC*: The specification is restricted to three soft constraints (`S5`, `S7`, `S8`) and is based on the solution which we initially submitted to the IHTC 2024.
- *5SC*: An extended specification implementing two additional soft constraints (`S1`, `S6`).
- *6SC*: A further extended specification additionally implementing soft constraint `S2`.
- *8SC*: The complete specification, modelling the full IHTP including all eight soft constraints.

Methodology. We report the best results our approach obtained with 10 min solving time for all 30 instances of the IHTC (cf. Table 1). We executed each of our approaches three times and report the best found solution over all runs. The table includes the four columns *3SC*, *5SC*, *6SC*, and *8SC*, which are marked to show which of our implementations achieved the respective result. For reference, we included the best reported solution[3] of the IHTC 2024 for every problem instance in the column *IHTC*. To enable a fair comparison across solutions, the cost values are calculated based on all eight soft constraints, regardless of the number of soft constraints considered in the implementation that obtained the best result. All evaluation runs were executed on a dedicated workstation equipped with an AMD EPYC 7763 64-Core processor and 256 GB of RAM.

Results. The *8SC* implementation achieved the best results for the first four instances (i01 to i04). The next 14 instances (i05 to i18) were best solved with either *6SC* or *5SC*, except i14, which was best solved by *3SC*. The remaining nine instances were mostly solved by *3SC* or *5SC*, except *i25*, which was best solved by *6SC*. In general, there is a trend whereby the larger instances were best optimised using the simpler implementations featuring fewer soft constraints.

[2] IHTP paper resources: https://github.com/Echtzeitsysteme/ihtp-paper-2026-resources (Accessed June 23, 2026)
[3] IHTC 2024 best results: https://ihtc2024.github.io/#section9 (Accessed June 25, 2026)

Table 1: Best found solutions of our approach for all 30 competition instances and a solve time limit of 10 min. The column *IHTC* describes the best value reported for the IHTC 2024, the column *cost* denotes the final achieved penalty cost of our implementation (lower is better), and a mark in the respective column shows which implementation achieved it.

			Implementation			
Instance	**IHTC**	**Cost**	**3SC**	**5SC**	**6SC**	**8SC**
i01	3,842	3,859				•
i02	1,264	1,317				•
i03	10,490	10,555				•
i04	1,884	2,748				•
i05	12,760	16,317			•	
i06	10,671	11,265		•		
i07	5,026	11,045			•	
i08	6,291	12,329		•		
i09	6,682	10,017			•	
i10	20,820	28,475		•		
i11	25,938	27,138		•		
i12	12,430	19,806			•	
i13	17,328	20,661			•	
i14	9,746	17,473	•			
i15	12,486	26,564		•		
i16	10,139	16,133		•		
i17	40,535	76,090		•		
i18	37,660	44,339		•		
i19	44,587	75,802	•			
i20	29,098	40,559	•			
i21	24,703	43,688	•			
i22	47,861	82,880	•			
i23	37,550	62,603	•			
i24	33,221	41,014	•			
i25	11,517	20,988			•	
i26	64,613	105,155	•			
i27	51,828	101,533		•		
i28	75,172	89,994	•			
i29	12,475	16,483		•		
i30	37,943	66,027		•		

Discussion. In theory, the quality of the solutions should improve with a higher number of implemented soft constraints. As a result, *8SC* should achieve the best result by outperforming all other implementations since it implements all of the specified soft constraints. However, since the time limit of 10 min clearly restricts the solver for the larger instances and a higher implemented number of soft constraints, the quality of the found solutions does not exceed the achieved costs of the simpler implementations, which leads to the presented result. Especially, all nurse-related soft constraints (`S2`, `S3`, and `S4`) substantially increase the MILP problem size. Removing them eliminates all soft constraints associated with the NPA, which greatly simplifies all other allocations. Among these three soft constraints, `S3` (continuity of care) and `S4` (nurse workload) proved to be particularly complex, as they require aggregated workload information and therefore introduce a cross-product over two virtual node types. This causes a significant rise in the number of MILP variables and associated constraints. Based on these observations, the reduced specification variant *5SC* achieves the best overall performance in the chosen setting. It consistently obtains feasible solutions for all instances, while, in most cases, outperforming the smaller *3SC* variant in solution quality. However, if we relax the time limitation and use the *8SC* GIPSL specification, we were able to calculate the *optimal* solution for two of the given IHTP problem instances (i01 with a cost of 3,842 and i03 with a cost of 10,490) as well as proving the optimality by using our GIPS-based approach.

In the IHTP, cost is computed as a weighted sum of penalties for violations of soft constraints (with changing weights across problem instances), making it hard to assess what the gap in quality of computed solution actually means for their comparative usability in practice. We, therefore, refrain from an interpretation.

6 Related Work

Casting relevant problems from the healthcare domain mathematically as optimisation problems and employing techniques from OR to solve these is an established field of research and various surveys review existing problems, solution techniques, or approaches to specific problems, e.g., [1, 2, 5, 31]. In this paper, we use the IHTP [14] as a use case to investigate in how far model-driven concepts can support specifying and solving optimisation problems in the healthcare domain. The IHTP problem is defined in [14]; 32 solutions have been submitted to the competition[4], a selection of which was presented at the EURO 2025 conference[5]. Besides introducing the problem, Ceschia et al. [14] also review the 32 competition submissions, classifying the used approaches as employing MILP, Constraint Programming (CP), and/or metaheuristics; the five finalists of the competition are presented in more detail. In the meantime, one of the five finalists published a detailed account of their approach [33]; they hybridise late acceptancy hill climbing with large neighbourhood search. To implement these, they employ a set of diverse search techniques, including exact methods like Mixed Integer Programming (MIP) and the computation of cliques of longest paths in a graph. Importantly, according to Ceschia et al. [14], who collect the best known results for the public and the hidden problem instances of the competition, so far, optimality has not been proved for any found solution for any of these problem instances, which we achieve for two instances in this paper; this also remains valid after the appearance of [33].

The main concern of our approach to the IHTC'24 problem, however, is *ease of specification*, where we aim to achieve a result that is modular, concise, and understandable for domain experts, but still mathematically precise. High-level constraint languages like Essence [19] or MiniZinc [29] exist and provide a range of desirable features: They are parametric, i.e., support defining an optimisation problem generically from which concrete instances can be automatically generated according to input data; they provide abstract data types like arrays or functions and support types. Still, a user is forced to specify an optimisation problem in terms of these abstract data types rather than being able to use terms of the

[4] International Healthcare Timetabling Competition 2024: https://ihtc2024.github.io (Accessed June 22, 2026)

[5] European Conference on Operational Research 2025: https://euro2025leeds.uk (Accessed June 22, 2026)

problem domain. In contrast, our approach is a specific instance of *model-driven optimisation* (see, e.g., [8, 21, 23, 38]), where the goal is to provide users with the possibility to specify optimisation problems via *domain models* on a higher level of abstraction, enabling them to express themselves in familiar, domain-specific terms and to integrate domain knowledge that might foster the search for optimal solutions. In our implementation, we use GIPS [17, 18, 24], (manually) translating the model- and OCL-based specification (see Section 3) into GIPSL (see Section 4), GIPS' internal specification language. Besides GIPS, tools that implement MDO include MDEOptimiser [11], MOMoT [8], or VIATRA-DSE [21]. While the general specification mechanisms for problems are similar to the one we present here, other MDO approaches use different concepts and algorithms to compute solutions. The most widely employed class of algorithms are evolutionary search—or similar meta-heuristic search procedures.

While we are not aware that MDO has been systematically applied in the healthcare domain, there exist approaches that are similar in spirit, i.e., other approaches that want to increase the accessibility of technical tools and methods by allowing healthcare experts to engage with these on a more abstract, less technical level and in terms they are familiar with in their domain. A—meanwhile slightly dated—survey on the use of UML for modelling in the healthcare domain has been performed by Vasilakis [36]. Caruso et al. developed a tool, called *CNL2ASP*, that translates specifications of optimisation problems written in a controlled natural language into an answer set programming specification and they evaluate their approach, amongst others, via two use cases from the healthcare domain, namely nurse scheduling and chemotherapy treatment scheduling [13]. Bernardi et al. develop a graphical DSL to model clinical pathways [6]; models in that language can be translated into formal models (different kinds of Petri nets), which can then be used to formally analyse the modelled pathways [6, 15]. Whether our approach actually enables domain experts to specify their optimisation problems—or at least offers a suitable format for joint development of these by domain and OR experts—remains to be evaluated.

7 Discussion and Future Work

In this paper, we explored how model-driven techniques can be applied to the specification and solving of complex optimisation problems in the healthcare domain, specifically the Integrated Healthcare Timetabling Problem (IHTP) from the Integrated Healthcare Timetabling Competition (IHTC) 2024. In doing so, we aimed to answer three research questions:

– **RQ1** *Can established declarative modelling formalisms—namely conceptual modelling using class diagrams, the OCL, and the declaration of model transformations—be used to express complex optimisation problems from the healthcare domain (in terms close to the domain)?* In Section 3, we presented a declarative specification of the IHTP using class diagrams to capture domain concepts, OCL expressions to define constraints and objective functions, and model transformations to characterise the available decision space. Class diagrams and OCL are well-established formalisms in the model-driven engineering community, which are widely used for conceptual-modelling purposes, like in our scenario. Using rule-based model transformations to declaratively specify the decisions that can be made to find valid and optimal solutions means constraints can be naturally integrated, minimising the search space by only allowing decisions that lead to valid solutions to be explored. In particular, the specifications shown in Section 3 use domain terminology throughout and encapsulate the technical aspects of the specification in a graphical format for easier accessibility.

– **RQ2** *Can model-based, declarative specifications be transferred to specifications in established, off-the-shelf solvers like Gurobi?* We briefly discuss in Section 4 how the declarative specification can be translated into a MILP formulation that can be solved by the off-the-shelf solver Gurobi. We have implemented this translation (though some steps currently remain manual) and used it for the experiments we report in Section 5.

– **RQ3** *Given a model-based specification and such a translation to a solver, can solutions be efficiently computed?* We report on experimental evaluation of our approach using the 30 problem instances provided by the IHTC in Section 5. Here, the picture is still more mixed: while we are able to solve all instances within the original time limit of 10 min, our approach is not yet competitive with the best-performing approaches in the competition. Improvements in efficiency and effectiveness of optimisation-problem solving typically rely on highly optimised problem formulations and bespoke pre-processing strategies, which require significant optimisation expertise and are often highly problem-specific. With our approach, we already obtain some of these 'for free' as constraints are embedded in transformation rules allowing GIPS to automatically discount many potential decisions without involving the general-purpose MILP solver and optimiser. This reduces the variable and constraint counts, which are key drivers of solver performance. We believe that we can integrate additional strategies in the GIPS pipeline to generate better problem formulations for the solver, without requiring significant additional optimisation expertise from the modeller, similar to related efforts such as the Conjure toolkit [4]. However, this remains to be explored in future work.

Currently, our approach requires a significant degree of manual translation of the initial problem specification before it can be processed by our GIPS tool. However, much of this translation follows a systematic set of steps, which we believe can be easily automated. In future work, we aim to develop a more automated pipeline for this process.

Other solutions submitted to the IHTC are just in the process of being formally published. We are currently studying these in detail to understand the specific strategies used by other teams. We are particularly interested in exploring whether some of the bespoke pre-processing strategies developed by other teams could be integrated into our approach to increase the number of strategies available without explicit coding.

Finally, we would like to undertake an empirical evaluation with clinical stakeholders to validate our approach in real-world scenarios. In particular, we are keen to understand the degree to which our approach to problem modelling enables participatory modelling and improves the explainability of optimisation outcomes.

References

[1] Zahraa A. Abdalkareem, Amiza Amir, Mohammed Azmi Al-Betar, Phaklen Ekhan, and Abdelaziz I. Hammouri. 2021. Healthcare scheduling in optimization context: a review. *Health and Technology* 11 (2021), 445−−469. Issue 3. doi:10.1007/s12553-021-00547-5

[2] Amir Ahmadi-Javid, Zahra Jalali, and Kenneth J. Klassen. 2017. Outpatient appointment systems in healthcare: A review of optimization studies. *European Journal of Operational Research* 258, 1 (2017), 3–34. doi:10.1016/J.EJOR.2016.06.064

[3] Mohamed A. Ahmed and Talal M. Alkhamis. 2009. Simulation optimization for an emergency department healthcare unit in Kuwait. *Eur. J. Oper. Res.* 198, 3 (2009), 936–942. doi:10.1016/J.EJOR.2008.10.025

[4] Özgür Akgün, Alan M. Frisch, Ian P. Gent, Christopher Jefferson, Ian Miguel, and Peter Nightingale. 2022. Conjure: Automatic Generation of Constraint Models from Problem Specifications. *Artificial Intelligence* 310 (2022), 103751. doi:10.1016/J.ARTINT.2022.103751

[5] Ali Ala and Feng Chen. 2022. Appointment Scheduling Problem in Complexity Systems of the Healthcare Services: A Comprehensive Review. *Journal of Healthcare Engineering* 2022, 1 (2022), 5819813. doi:10.1155/2022/5819813

[6] Simona Bernardi, Cristian Mahulea, and Jorge Albareda. 2019. Toward a decision support system for the clinical pathways assessment. *Discrete Event Dynamic Systems* 29, 1 (2019), 91–125. doi:10.1007/S10626-019-00279-9

[7] Bogdan Bichescu, Randy V. Bradley, David D. Dobrzykowski, Iana Shaheen, and Antoinette Smith. 2025. Examining the use of analytics in healthcare operations management: A systematic and narrative literature review. *Decision Sciences* 56, 5 (July 2025), 449–470. doi:10.1111/deci.70011

[8] Robert Bill, Martin Fleck, Javier Troya, Tanja Mayerhofer, and Manuel Wimmer. 2019. A local and global tour on MOMoT. *Software & Systems Modeling* 18, 2 (2019), 1017–1046. doi:10.1007/S10270-017-0644-3

[9] Ilhem Boussaïd, Patrick Siarry, and Mohamed Ahmed-Nacer. 2017. A survey on search-based model-driven engineering. *Automated Software Engineering* 24, 2 (April 2017), 233–294. doi:10.1007/s10515-017-0215-4

[10] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. 2017. *Model-Driven Software Engineering in Practice, Second Edition.* Morgan & Claypool Publishers. doi:10.2200/S00751ED2V01Y201701SWE004

[11] Alexandru Burdusel, Steffen Zschaler, and Daniel Strüber. 2018. MDEoptimiser: a search based model engineering tool. In *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS 2018, Copenhagen, Denmark, October 14–19, 2018*, Önder Babur, Daniel Strüber, Silvia Abrahão, Loli Burgueño, Martin Gogolla, Joel Greenyer, Sahar Kokaly, Dimitris S. Kolovos, Tanja Mayerhofer, and Mansooreh Zahedi (Eds.). ACM, 12–16. doi:10.1145/3270112.3270130

[12] Jordi Cabot and Martin Gogolla. 2012. *Object Constraint Language (OCL): A Definitive Guide.* Springer Berlin Heidelberg, Berlin, Heidelberg, Chapter 3, 58–90. doi:10.1007/978-3-642-30982-3_3

[13] Simone Caruso, Carmine Dodaro, Marco Maratea, Marco Mochi, and Francesco Riccio. 2024. CNL2ASP: Converting Controlled Natural Language Sentences into ASP. *Theory and Practice of Logic Programming* 24, 2 (2024), 196–226. doi:10.1017/S1471068423000388

[14] Sara Ceschia, Roberto Maria Rosati, Andrea Schaerf, Pieter Smet, Greet Vanden Berghe, and Eugenia Zanazzo. 2025. The Integrated Healthcare Timetabling competition 2024. *Operations Research, Data Analytics and Logistics* 45 (2025), 200481. doi:10.1016/j.ordal.2025.200481

[15] Daniel Clavel, Cristian Mahulea, and Manuel Silva. 2019. From Healthcare System Specifications to Formal Models. In *2019 IEEE International Conference on Systems, Man and Cybernetics, SMC 2019, Bari, Italy, October 6–9, 2019.* IEEE, 2344–2351. doi:10.1109/SMC.2019.8914654

[16] Emma Coles, Julie Anderson, Margaret Maxwell, Fiona M. Harris, Nicola M. Gray, Gill Milner, and Stephen MacGillivray. 2020. The influence of contextual factors on healthcare quality improvement initiatives: a realist review. *Systematic Reviews* 9, 1 (April 2020), 94. doi:10.1186/s13643-020-01344-3

[17] Sebastian Ehmes. 2025. *GIPS: A Graph- and MILP-based Optimization Framework.* Ph. D. Dissertation. Technische Universität Darmstadt, Darmstadt. doi:10.26083/tuprints-00030156

[18] Sebastian Ehmes, Maximilian Kratz, and Andy Schürr. 2022. Graph-Based Specification and Automated Construction of ILP Problems. In *Proceedings of the Thirteenth International Workshop on Graph Computation Models, GCM@STAF 2022, Nantes, France, 6th July 2022 (EPTCS, Vol. 374)*, Reiko Heckel and Christopher M. Poskitt (Eds.). Open Publishing Association, 3–22. doi:10.4204/EPTCS.374.3

[19] Alan M. Frisch, Warwick Harvey, Christopher Jefferson, Bernadette Martínez Hernández, and Ian Miguel. 2008. Essence: A constraint language for specifying combinatorial problems. *Constraints* 13, 3 (2008), 268–306. doi:10.1007/S10601-008-9047-Y

[20] Wei Gu, Xin Wang, and S. Elizabeth McGregor. 2010. Optimization of preventive health care facility locations. *International Journal of Health Geographics* 9 (2010), 17. doi:10.1186/1476-072X-9-17

[21] Ábel Hegedüs, Ákos Horváth, and Dániel Varró. 2015. A model-driven framework for guided design space exploration. *Automated Software Engineering* 22, 3 (2015), 399–436. doi:10.1007/S10515-014-0163-1

[22] Frederick S. Hillier and Gerald J. Lieberman. 2010. *Introduction to Operations Research* (9 ed.). McGraw-Hill Education. http://www.maths.lse.ac.uk/Personal/stengel/HillierLieberman9thEdition.pdf Accessed 8 August 2025.

[23] Stefan John, Jens Kosiol, Leen Lambers, and Gabriele Taentzer. 2023. A graph-based framework for model-driven optimization facilitating impact analysis of mutation operator properties. *Software and Systems Modeling* 22, 4 (Jan. 2023), 1281–1318. doi:10.1007/s10270-022-01078-x

[24] Maximilian Kratz, Sebastian Ehmes, Philipp Maximilian Menzel, and Andy Schürr. 2025. Model-Driven Rapid Prototyping for Control Algorithms with the GIPS Framework (System Description). In *Proceedings of the Fourteenth and Fifteenth International Workshop on Graph Computation Models, GCM 2024, Leicester, UK and Enschede, the Netherlands, 18 July 2023 and 9 July 2024 (EPTCS, Vol. 417)*, Jörg Endrullis, Dominik Grzelak, Tobias Heindel, and Jens Kosiol (Eds.). 157–172. doi:10.4204/EPTCS.417.9

[25] Maximilian Kratz, Jule Pfau, Steffen Zschaler, Jens Kosiol, and Andy Schürr. 2025. High-Level Specification of MILP Problems with Class Diagrams and Graph Transformations: A Case Study in Model-Driven Optimisation of the Integrated Healthcare Timetabling Problem. doi:10.2139/ssrn.5768119 Pre-print.

[26] Maximilian Kratz, Steffen Zschaler, Jens Kosiol, and Gabriele Taentzer. 2025. Automatic Generation of Combinatorial Reoptimisation Problem Specifications: A Vision. *CoRR* abs/2510.02002 (2025). arXiv:2510.02002 doi:10.48550/ARXIV.2510.02002

[27] Joaquim R. R. A. Martins and Andrew Ning. 2021. *Engineering Design Optimization.* Cambridge University Press. doi:10.1017/9781108980647

[28] Giacomo Zambelli Michele Conforti, Gérard Cornuéjols. 2014. *Integer Programming.* Springer Cham. doi:10.1007/978-3-319-11008-0

[29] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. 2007. MiniZinc: Towards a Standard CP Modelling Language. In *13th International Conference Principles and Practice of Constraint Programming (CP'07) (Lecture Notes in Computer Science, Vol. 4741)*, Christian Bessiere (Ed.). Springer Berlin Heidelberg, 529–543. doi:10.1007/978-3-540-74970-7_38

[30] Muhammed Ordu, Eren Demir, Chris Tofallis, and Murat M. Gunal. 2021. A novel healthcare resource allocation decision support tool: A forecasting-simulation-optimization approach. *Journal of the Operational Research Society* 72, 3 (2021), 485–500. doi:10.1080/01605682.2019.1700186

[31] Abdur Rais and Ana Viana. 2011. Operations Research in Healthcare: a survey. *Int. Trans. Oper. Res.* 18, 1 (2011), 1–31. doi:10.1111/J.1475-3995.2010.00767.X

[32] Real-Time Systems Lab, TU Darmstadt. 2025. GIPS (GitHub Repository). https://github.com/Echtzeitsysteme/gips. accessed 10 August 2025.

[33] Lisa Garcia Tercero, David Van Bulck, Karel Devriesere, Samuel Bakker, and Dries R. Goossens. 2026. A multi-neighborhood hybrid approach for the integrated healthcare timetabling competition 2024. *Comput. Oper. Res.* 191 (2026), 107453. doi:10.1016/J.COR.2026.107453

[34] Stefan Tomaszek. 2021. *Modellbasierte Einbettung von virtuellen Netzwerken in Rechenzentren.* Ph. D. Dissertation. Technical University of Darmstadt, Darmstadt. doi:10.12921/tuprints-00017362

[35] Stefan Tomaszek, Erhan Leblebici, Lin Wang, and Andy Schürr. 2018. Virtual Network Embedding: Reducing the Search Space by Model Transformation Techniques. In *Theory and Practice of Model Transformation*, Arend Rensink and Jesús Sánchez Cuadrado (Eds.). Springer International Publishing, Cham, 59–75. doi:10.1007/978-3-319-93317-7_2

[36] Christos Vasilakis, Dorota Lecnzarowicz, and Chooi Lee. 2008. Application of Unified Modelling Language (UML) to the Modelling of Health Care Systems: An Introduction and Literature Survey. *International Journal of Healthcare Information Systems and Informatics (IJHISI)* 3, 4 (2008), 39–52. doi:10.4018/JHISI.2008100103

[37] Laurence Wolsey. 2020. *Integer Programming: 2nd Edition.* John Wiley & Sons, Ltd. doi:10.1002/9781119606475

[38] Steffen Zschaler and Lawrence Mandow. 2016. Towards Model-Based Optimisation: Using Domain Knowledge Explicitly. In *Software Technologies: Applications and Foundations – STAF 2016 Collocated Workshops: DataMod, GCM, HOFM, MELO, SEMS, VeryComp, Vienna, Austria, July 4–8, 2016, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 9946)*, Paolo Milazzo, Dániel Varró, and Manuel Wimmer (Eds.). Springer, 317–329. doi:10.1007/978-3-319-50230-4_24