A KNOWLEDGE REPRESENTATION APPROACH TO AUTOMATED MATHEMATICAL MODELLING

Bahadorreza Ofoghi
School of Information Technology
Deakin University
Burwood, VIC 3125 Australia
b.ofoghi@deakin.edu.au

Vicky Mak
School of Information Technology
Deakin University
Burwood, VIC 3125 Australia
vicky.mak@deakin.edu.au

John Yearwood
School of Information Technology
Deakin University
Burwood, VIC 3125 Australia
john.yearwood@deakin.edu.au

November 13, 2020

ABSTRACT

Mathematicians formulate complex mathematical models based on user requirements to solve a diverse range of problems in different domains. These models are, in most cases, represented through several mathematical equations and constraints. This modelling task comprises several time-intensive processes that require both mathematical expertise and (problem) domain knowledge. In an attempt to automate these processes, we have developed an ontology for Mixed Integer Linear Programming (MILP) problems to formulate expert mathematician knowledge and in this paper, we show how this new ontology can be utilized for modelling a relatively straightforward MILP problem, a Machine Scheduling example. We also show that more complex MILP problems, such as the Asymmetric Travelling Salesman Problem (ATSP), however, are not readily amenable to simple elicitation of user requirements and the utilization of the proposed mathematical model ontology. Therefore, an automatic mathematical modelling framework is proposed for such complex MILP problems, which includes a problem (requirement) elicitation module connected to a model extraction module through a translation engine that bridges between the non-expert problem domain and the expert mathematical model domain. This framework is argued to have the necessary components to effectively tackle the automation of modelling task of the more intricate MILP problems such as the ATSP.

Keywords Optimization · Automatic modelling · Ontology

1 Introduction

In the last few decades, researchers in the field of Combinatorial Optimization (CO) have formulated mathematical models and developed solution algorithms to automate otherwise time-consuming and labour-intensive processes. Not only did these automations save end-users’ time, but also and more importantly, they brought huge social and economic benefits. For example, the San Francisco Police Department implemented a Integer Linear Programming (ILP)-based support system for deploying patrol officers and saved 14 million USD a year; PSI Insurance developed a series of optimization-based models, including ILP, to value and trade mortgage-based securities, which resulted in a 10 billion USD increase in trading volume; P&G developed ILP and network models to improve work processes and saved 200 million USD [8].

There are two major components in mathematical optimization modelling: i) the formulation of a mathematical model that captures the underlying business requirements, and ii) the computation required to solve the problem based on the mathematical model. A modern-day computer can perform the second task reasonably well. However, at this point in time, only properly trained human optimization experts are able to perform the first task. This research is to investigate how and to what extent a computer can carry out the first task in place of a human. In this paper, we discuss the development of a first Mixed-Integer Linear Programming (MILP) ontology that formulates expert knowledge in the domain with the aim of automating MILP modelling. We will show that for straightforward MILP cases this ontology
in combination with brief user requirement gathering is sufficient for automated modelling; however, for more complex
problems such as the Asymmetric Travelling Salesman Problem (ATSP), a more comprehensive framework is proposed,
one that consists of a problem elicitation module and a modelling module. While the former directly interacts with
end-user to elicit problem specifications, the latter is responsible for mathematical modelling. The two modules are
connected with each other through a translation engine so the problem specifications can be translated into a formal
mathematical model.

1.1 Combinatorial Optimization and Mixed-Integer Linear Programming

Combinatorial Optimization Problems (COPs) arise in many real-life applications such as scheduling, planning,
resource allocation, routing, and time-tableing. COPs can be solved optimally with exact algorithms or quickly (but
not necessarily optimally) using heuristics. Heuristic methods do not guarantee optimality; thus, when exact solutions
matter, e.g., in medical applications where any improvement in the objective function means more lives saved, the
exact algorithms will always be preferred. Such algorithms essentially implement an exhaustive search with algebra
as the theoretical basis like in the case of MILP or logical inferences such as in the case of Constraint Programming
(CP). CP and MILP each has its strengths and weaknesses. The two top-performing algorithms in the 2nd Nurse
Rostering Competition are MILP-based methods [7]. MILP is a very practical tool in modelling real-life COPs; even
some nonlinear terms (such as quadratic binary and bilinear terms) can be linearized through remodelling. For these
reasons, we focus our study on the automatic modelling of COPs using MILP in this paper. A MILP is given as min (or
max)\{c \cdot x + d \cdot y : Ax + By \leq f, x \in \mathbb{Z}_{n}^{m}, y \in \mathbb{R}_{n}^{m}\}, with x = (x_1, \ldots, x_n) and y = (y_1, \ldots, y_p) the
decision variables; c and d the cost coefficients; for c a n-vector of \mathbb{Z}, d a p-vector of \mathbb{R}; A \in \mathbb{Z}_{m \times n} and B \in \mathbb{R}_{m \times p} the
constraint coefficient matrices; and f a m-vector of \mathbb{Z}.

The “translation” required from an end-user’s description of the problem into a proper MILP formulation is currently
something only a trained human expert MILP modellers are capable of performing. One key component of these general
models is the knowledge representation of COPs and MILPs. We will review some related work in this direction in the
next section.

1.2 Related work

Current automation in mathematical modelling is mainly focused around automatic model selection, e.g., [12][10] and
Learning by Examples (solutions), e.g., [3][18]. However, in the context of generalized mathematical modelling, these
methods fall short as they are mainly concerned with parameterized model families.

There has been previous work in the area of constraint or model acquisition for Constraint Programming problems [3][2]
and in model acquisition for Integer Linear Programming [18]. However, the test cases in these studies are of small
scales and in some cases, knowledge about the full set of feasible solutions is required in advance. The (automated)
task of constraint or model acquisition based on passive or active learning from positive versus negative examples can
be achieved by exhaustive testing within a list of candidate constraints or through the application of logic constraint
inference from historical data [13]. Neither of these approaches is suitable for MILP problems. The reason is, in
most cases, end-users do not have access to any feasible solutions (positive examples) or infeasible solutions (negative
examples)—even determining the existence of positive examples in most real-life CO problems is NP-complete (e.g.,
constrained routing problems). In the absence of a full set of feasible solutions, invalid constraints are likely to be
reached. Even with the full set of feasible solutions, inferring a MILP model implies finding a complete polyhedral
description of solution points which results in a computation time that is exponential to the number of variables.

The task of automatic model selection and machine learning has similarities with constraint acquisition for CP problems
considering the fact that given a set of historical data, several models can be tested with reference to a known evaluation
schema [20]. The works in the domain of automatic model selection have been mainly concerned with parametrized
model families. One recent example is Auto-Weka 2.0 [12] that employs a Bayesian optimization technique to
search through the space of Weka’s machine learning algorithms and their relevant hyperparameters to select the best-
performing model given a specific data set. Auto-sklearn [10] also makes use of Bayesian optimization to implement
a similar model search and selection technique. Auto-sklearn, however, employs additional ensemble construction
techniques and meta-learning. The latter is concerned with the utilization of meta-knowledge to find an effective
mapping between problem characteristics and algorithm performances [4].

Automatic solution of Mathematical Word Problems (MWP) is another closely related domain to the task of automated
MILP modelling. Proposed methods to solve automatic MWP include rule-based (symbolic) artificial intelligence
techniques, feature engineering and selection using machine learning techniques, and deep learning [22]. These methods,
however, approach the solution of MWP by starting from a pre-existing textual description of the problem, hence word
problems. While this assumption works well for applications such as those with educational purposes, in the context of
real-world business solution offering the full description of the problems is not a given. Instead, problem elicitation via interaction with end-users will result in the problem description.

The above challenges and shortcomings in automated mathematical modelling using learning by examples, automatic model selection techniques, and automatic MWP solution necessitates the development of a knowledge-based semantic approach that can well generalize at least to a specific group of mathematical constraints and solutions within the sub-domain of COPs and MILP models. These models are explained in more detail in the next section.

The OntoMathPro ontology [17] is an OWL-based resource that encapsulates a large number of mathematical concepts and the semantic relationships among them. The ontology covers a wide range in the mathematics domain that could benefit novice students to expert mathematicians for information extraction as well as learning and semantic search of formulas. Mocassin Ontology [19] is another ontology that has a focus on mathematical information extraction from scholarly articles in the domain, with a focus on the structure of the mathematics-related articles rather than the abstract mathematical concepts and relations.

OMDoc [11] is an extensible XML-based mathematical terminology and language that implements mathematical modelling markups at the three levels of theory, statement, and object. OMDoc makes use of MathML [6] and OpenMath [5] objects and definitions of symbols to specify mathematical notations. The OMDoc OWL ontology captures semantic relationships such as whole-part, logical dependency, and verbalizing properties, the latter of which are specifically used to reference other OMDoc elements and provide inline definitions of phrase-level constructs.

Natural Sciences and Technology [9] is an ontology of mathematical concepts and relations developed in the Russian language, especially covering high-school and novice university-level mathematics. This ontology has a specific focus on information retrieval and text analysis in the mathematics domain and implements relations such as is-a and whole-part as well as dependence or associations.

ScienceWISE [1] has put together a series of scientific articles and concepts in several domains, including mathematics, that are related to each other via a set of ontological relations. This ontology was initially formed on the basis of information available from online encyclopedias and other domain-specific resources. The relations that connect ScienceWISE concepts are both generic/taxonomic (such as is-a and part-whole) and domain-specific (e.g., is a model of). Cambridge Mathematical Thesaurus [21] is another such ontology, which mostly covers mathematical concepts at the undergraduate level with relations such as dependence and associations. With a focus on education, this ontology has entries in several languages.

1.3 Outline and contributions

In this work, we focus on the design and development of an automatic modeller to deal with COPs as a specific mathematical modelling task. Our main contributions, thus, include the following:

- The development of the first MILP ontology that expresses mathematical expert knowledge. We will show that this ontology alone can be used to eliminate/minimize the expert’s involvement in the modelling of straightforward MILP problems.
- The development of a new automated mathematical modelling framework for MILP problems, which we will show is necessary in the case of more complex MILP problems.

The rest of the paper is organized as follows. In the next section, the new MILP ontology will be introduced, this is then followed by the descriptions of how the automated modelling of a straightforward MILP problem (with a machine scheduling example) can be achieved using the proposed ontology. Then, a more complex MILP problem (i.e., the Asymmetric Travelling Salesman Problem) will be discussed, the several intricacies of which, we will argue, will necessitate the need for the more comprehensive framework of MILP modelling automation. The main elements of the proposed framework for the same modelling task will be defined next. We will conclude the paper in the last section.

2 Automated MILP modelling

In this section, we introduce the MILP ontology we have developed and then, we use a simple machine scheduling problem as an example to explain the process of MILP model formulation by a human modeller from a user description to a mathematical model. We will show how this process can benefit from the proposed MILP ontology to become automated. We will then discuss the necessity for a more complex framework for automated modelling of more complex MILP problems such as the ATSP.
2.1 The MILP ontology

We propose the following ontology for MILP models as shown in Figure 1. The ontology has a top-level class for MILP that has Problem Sense, Objective Function, and Constraint as the main parts.

An individual instance in the class Constraint has a Left-Hand-Side (LHS) and a Right-Hand-Side (RHS). The LHS is a linear function of decision variables and the RHS is a scalar constant. The Linear Function is a sum of a multiple of coefficients and decision variables. The relevant entities are Coefficients, Operators, and Decision Variables. Instances of Operator are + and ×. Both Decision Variable and Coefficient have Number Type and Index Set.

A decision variable or a coefficient can have multiple indices and each has a set associated with it. E.g., a set of binary decision variables $x_{i,j,k}$ may have $i = 1, \ldots, 10$ that represents the ID numbers of a set of jobs, $j \in J$ for $J$ the set of 12 months in a year, and $k \in K$ the set of staff members. The binary variable may represent whether or not Staff $k$ is to perform Job $i$ in Month $j$.

The Iterator is similar to the Index Set – the former enumerates the set of constraints in a MILP and the latter enumerates the set of variables in a MILP. The class Problem Sense has instances max and min. The class Sense of the constraints has instances $\leq$, $=,$, and $\geq$. The Objective Function is also a linear function of decision variables, with “costs” of the variables as coefficients.

2.1.1 Implementation

The proposed MILP ontology was developed using Protege [16] and the Web Ontology Language (OWL). The rdfs:subClassOf relation was used within the OWL representation of the ontology to define the parent-child relations (i.e., is-a) between the different above-mentioned classes, e.g.,

```xml
<SubClassOf>
  <Class IRI="#SetCovering"/>
  <Class IRI="#Constraint"/>
</SubClassOf>
```

that shows the class SetCovering is a sub class of Constraint.

The part-whole relations (i.e., has-a) are not directly supported within Protege; however, such relations were defined as object properties for the relevant classes;

```xml
<Declaration>
  <ObjectProperty IRI="#part_of"/>
</Declaration>
```

e.g.,
which expresses the part-whole relation between class Sense and Constraint, the former being a part of the latter.

2.1.2 Constraint SubClasses

One may think that COPs are wide and broad so it will not be scalable to maintain a problem repository. Despite the large number of variations of COPs, however, there are only a relatively small number of types of constraints. Many scheduling and routing problems have constraints in common due to the fact that the underlying requirements are the same. For example, allocating jobs to machines are the same as allocating customers to delivery vehicles, both may have precedence relations or time-windows constraints. A MILP may contain exponentially (but finitely) many constraints but the number of classes of constraints is usually reasonably small. In this section, we present a number of commonly used constraints. In what follows, for simplicity, we use $a \cdot x$ to represent the dot product $< a \cdot x >$. Let $x \in \{0, 1\}^n$ be a set of $n$ binary variables. The Set Covering, Set Partitioning, and Set Packing constraints are given by

\[
\begin{align*}
ax & \geq 1 \quad (1) \\
ax & = 1 \quad (2) \\
ax & \leq 1 \quad (3)
\end{align*}
\]

respectively, for $a \in \{0, 1\}^n$. (2) and (3) are different in modelling though mathematically they are equivalent as one can be transformed to another with the help of additional variables.) If the right hand side for (1) and (2) are $Z$, $a \in \{0, 1, -1\}$ then we have the Weighted Set Covering and Weighted Set Partitioning respectively, and if on top of that, $a \in \mathbb{Q}_+^n$, then we have the Generalized Set Covering and Generalized Set Partitioning respectively. If $a \in \mathbb{Q}_+^n$, $b \in \mathbb{Z}_+$, and $x \in \mathbb{Q}_+^n$, then $ax \leq b$ is a Knapsack Constraint. If $x \in \{0, 1\}^n$, we have a 0-1 Knapsack Constraint. The following constraints are used for formulating logical relations.

- **Either-or** condition. Suppose we wish to have either $f(x) \leq 0$ or $g(x) \leq 0$, we use the constraints $f(x) \leq Mt$ and $g(x) \leq M(1 - t)$, for $f(x)$ and $g(x)$ linear functions of decision variables $x = (x_1, \ldots, x_n)$, $M$ a sufficiently larger number, and $t$ a binary decision variable.

- **If-then** condition. Suppose $g(x) \leq 0$ will be true if $f(x) > 0$ is true, we can model such a condition by using constraints $g(x) \leq Mt$ and $f(x) \leq M(1 - t)$, again, for $x = (x_1, \ldots, x_n)$, $M$ a sufficiently large number, and $t$ a binary decision variable.

The *if-then* constraints are one of the most commonly used constraint class in modelling logical relations, for example, like the ones below.

- If we produce Product A, then we must also produce at least two of Products B, C, D, or E;
- If a class is not scheduled at a particular time, then no students will be allocated to this time;
- A occurs if and only if all of B, C, D, and E occurs.

A few classic examples are provided below.

- To model “if $f(x)$ occurs ($f(x) = 1$), then $g(x)$ occurs ($g(x) = 1$)””, for $x \in \{0, 1\}^n$ is a set of binary decision variables, $f(x)$ and $g(x)$ linear functions of $x$ with $f(x), g(x) \in \{0, 1\}$, we have that

\[
f(x) \leq g(x)
\]

- Let $x_A$ be a binary decision variable with $x_A = 1$ if Task A is to be carried out, $x_A = 0$ otherwise; and $x_{B_j}$ for $j = 1, \ldots, n$ be binary variables with $x_{B_j} = 1$ is Task $B_j$ to be carried out, $x_{B_j} = 0$ otherwise. To model A occurs if all of $B_1, \ldots, B_n$ occur, we have that:

\[
x_{B_1} + \cdots + x_{B_n} \leq n - 1 + x_A
\]

- For $A$ occurs only if all of $B_1, \ldots, B_n$ occur, we have that

\[
x_A \leq x_{B_j}, \quad \forall j = 1, \ldots, n
\]
• For A occurs if and only if all of B₁, . . . , Bₙ occur, we use (5) and (6) together.
• If A occurs then f(x) = C, for C ∈ ℚ. We have that:
\[-M(1 - z_A) + f(x) \leq C\]
\[M(1 - z_A) + f(x) \geq C\] (7)

2.2 A machine scheduling example

Suppose that a business owner describes his decision making problem as follows: I would like to allocate 100 jobs to 10 machines in the most cost-effective way. All jobs can be performed on all machines and each job must be done once. It needs to be decided which machine should be allocated to each job. A job can be performed on 1 machine only. Each machine can perform no more than 10 jobs. The business owner records the costs of performing each job on each machine in a spreadsheet which will be used as input data for the decision-making problem. In the case of this machine scheduling example, given the straightforward problem definition, a direct (textual) parsing and rule-based mapping from the problem specification to the MILP ontology concepts and instantiation of individuals results in the mathematical model. This is specifically possible as the problem specifications are well-categorized and are possible to be directly associated with the specific MILP ontology concepts. From here, the automated modelling occurs by traversing the MILP ontology relations (using the OWL reasoner) to combine and formulate the mathematical ingredients of the MILP model (e.g., the elements in the problem description that are index sets comprise the Constraints of the model through part_of relations). The formulation of an MILP for the machine scheduling problem using IBM ILOG OPL, a general purpose modelling language.

1. The number 100 and the word jobs trigger the enumeration of the index set range Jobs = 1..100;
2. Similarly, 10 and machines trigger the enumeration of the index set range Machines = 1..10;
3. The word cost triggers the request for input data, for each job and each machine, the cost of performing the job on the machine Costs[Jobs][Machines];
4. The decision is a yes/no to allocate a job to a machine, so dvar Boolean allocate[Jobs][Machines] is enumerated for each job-machine combination;
5. The words allocate . . . in the most cost-effective way trigger the calculation of the total cost of job-machine allocations, which is the sum of costs of performing all jobs on the machines where the job-machine allocation has a “yes” answer, i.e., sum(j in Jobs, m in Machines) Costs[j][m]*allocate[j][m], giving us a linear function as the objective function;
6. The words in the most cost-effective way also indicate that the sense of the objective is minimize;
7. Each job is to be performed by exactly 1 machine, which is equivalent to “out of all machines, allocate 1”, iterated over all Jobs i.e.,
\[\text{sum(m in Machines) allocate[j][m] == 1;}\]
8. A straightforward translation of “each machine is to perform no more than 10 jobs” to “total number of jobs allocated to each machine is no more than 10”, iterated over all Machines,
\[\text{sum(j in Jobs) allocate[j][m] <= 10.}\]

2.3 The Asymmetric Travelling Salesman Problem

The Asymmetric Travelling Salesman Problem (ATSP) is a well studied COP. A business owner may describe his decision-making problem as follows: I have to deliver some goods to 100 customers located in different cities, and I have to do this on my own. I need to find the most cost-effective way to travel to all the customers. The business owner has the cost of travel between each pair of cities recorded in a spreadsheet. Since he did not describe any other restrictions, such as a time-window for each customer, we assume that there are no other constraints. If the costs of travel from one city to another in different directions are different, this is an ATSP. This time, with a direct parsing from the NLP description of the end-user, one may only get as far as enumerating the set of customers/cities (\{String\} Customers;) and the distance matrix (int Distances[Customers][Customers]). The rest of the model is far beyond what a non-expert end-user is able to describe.

First of all, unlike the case of the machine scheduling example where the decision was straightforward — the allocation of jobs to machines, the decision variables for the ATSP use a binary decision variable to represent each ordered pair of customers (dvar boolean x[Customers][Customers])—not a trivial concept for a non-expert to grasp. The objective function is to minimize the sum of costs of travel between customers in the ordered pairs that represent the sequence of travel in a solution (minimize sum {i,j in Customers, i != j} Cost[i][j] * x[i][j]).
The well-known MILP formulation for the ATSP has the following sets of constraints: i) the Assignment Constraints (AC) formed by two sets of Set Partitioning constraints, and ii) the Subtour Elimination Constraints (SEC).

The assignment constraints are as follows:

\[
\forall i \in \text{Customers} \{ \\
\sum_{j \in \text{Customers}, j \neq i} x[i][j] = 1; \\
\sum_{j \in \text{Customers}, j \neq i} x[j][i] = 1; \\
\}
\]

These constraints enforce that each customer must have one that precedes it and one that succeeds it, (so the traveller’s home is the one that precedes the first customer and succeeds the last customer in the optimal sequence) whereas an end-user would normally just say that they have to visit each customer (exactly) once.

The objective function and the ACs above form a well-known COP called the Assignment Problem (AP). The solutions to an AP may induce multiple disjoint cycles called “subtours” which are infeasible for the ATSP, and therefore, SECs are required. This means that a full model cannot be directly elicited from the user and a knowledge-based model construction mechanism is required.

Commonly used SECs include exponential-size Set Covering version and Weighted Set Packing version as well as polynomial-size Miller-Tucker-Zemlin (MTZ) If-then Constraint version. The former has a stronger Linear Programming relaxation but requires sophisticated solution methodology, with SECs generated “on the fly” as they cannot be enumerated completely for large-scale problems, whereas the latter is much easier to implement. The MTZ SEC requires enumerating another set of variables to represent the time of arrival at a customer, and a set of If-then Constraints to model the logic condition “if Customer \( j \) is visited immediately after Customer \( i \), then \( \text{time}[j] \geq \text{time}[i] + 1 \).” This leads to another important question, which SEC or which full model should be used when more than one is available. In the example provided above, as there are only 100 customers, given the advancements of modern MILP solvers such as IBM CPLEX and Gurobi, the polynomial-size model is well capable of solving the ATSP to optimality in a reasonable amount of time in practice.

2.4 Modelling challenges

To summarize the modelling challenges, in the ATSP example, the decision variables cannot be parsed in a straightforward manner. First of all, a translation for “each element to occur exactly once” to “each element must have one that precedes it and one that succeeds it” for all sequencing problems is needed. Fortunately, we know for a fact that scheduling problems and routing problems have many requirements in common. Similarly for many other common features in different families of COPs. A translation engine is needed to handle the translation of a number of commonly used standard COP requirements into their corresponding standard MILP formulations, and in the case of the ATSP, to handle the need of the SECs. This can be facilitated through expert knowledge represented in a knowledge-base as well as the storage and retrieval of previously known, similar problem specifications in order to translate a problem description to a proper COP definition, and then to a MILP formulation.

Secondly, there are problem specifications for which multiple MILP formulations can be realized. In such cases, a direct formulation of the model from the problem description by end-user alone will not suffice. Multiple models will need to be evaluated for their accuracy and relevance using prescribed evaluation metrics to rank and select the best-suited model, as is the case with ATSP.

The third challenge is even more difficult to address. Consider the decision problem of sequencing a multi-leaf collimator (MLC) in the step-and-shot intensity-modulated radiotherapy treatment planning. At each treatment angle, a number of MLC apertures are required to be formed for radiation to be delivered through these apertures. The changeover time between one aperture to another depends on the movements required for the MLC leaves. The objective is to find the sequence of MLC apertures such that the total changeover times between apertures is minimized. The automatic modeller should be able to tell that this is again a sequencing problem for a set of elements, each to appear once, with the total “cost of ordered pairs” minimized, and hence be able to recognize that this is in fact an ATSP.

3 The proposed framework for automated mathematical modelling

The overall structure of the proposed framework is shown in Figure 2. In which there are two major modules. The Elicitation module has direct interactions with the non-expert end-user to elicit the problem specifications. The fundamentals of Human Computer Interactions (HCI) will make it possible for this module to effectively engage with the end-user and identify the necessary elements to develop a full definition of the user problem. The problem repository
of the framework maintains historical problem definitions for future use while the problem knowledge-base is consulted for more formal generation of the problem definition.

The modelling module is connected with the elicitation module through the translation engine which is an intermediary to make it possible for the model construction engine to extract and combine the model elements. Model construction requires expert knowledge formulated within a model ontology, the details of which will be described in the following sections. In some cases, several models may be generated by the model construction engine. The model evaluation engine is responsible for evaluation and ranking of these models.

Figure 2: The overall structure of the proposed automated mathematical modelling framework.

3.1 Problem elicitation module

A problem elicitation engine will elicit a proper definition of the COP from a non-expert end-user via HCI, in the same way a human modeller would. COPs are not as open-ended as one might think. While a MILP may contain many constraints, the number of constraint classes is much smaller. COPs typically seek decisions on if/how/how many something is to occur, the assignment of objects, and the logical relationships between events/entities; e.g., temporal sequencing. Allocating nurses to shifts, or customers to vehicles are “assignments” with different context/complexity for each problem. The order in which customers are served and sequencing of flight legs, for instance, fall under “sequencing”. We believe there is a reasonably small, finite number of constraints in terms of the nature of the requirement that the COP models, and similarly for the objectives. For this reason, the problem repository is expected to be scalable. The challenge is how to represent, store, retrieve the knowledge of the commonly used COP features and translate them to the appropriate constraint classes, and use this knowledge to guide the end-users to articulate all the required business requirements.

3.2 Modelling module

The modelling module will be composed of a MILP Ontology as presented earlier, a Model Construction Engine, and a Model Evaluation engine.

There are two possible outcomes from the elicitation module: i) a full model (or may be even more than one model) that captures all problem descriptions does exist in the Problem Repository, and ii) a full model does not exist but there exist one or more partial models each capturing some subsets of the problem description. In the case of the latter, there are two approaches to construct a full model: i) select one as basis and construct new elements (model synthesis), in which case, an evaluation schema is required to rank the partial models by their mathematical structure; ii) combine multiple partial models to produce a full model, in which case, a model fusion algorithm is required.

Evaluation metrics are required at different levels. In the case of partial models: i) the amount of problem description a partial model captures; and ii) whether the partial model is easy to implement or to “fuse” with new components. For example, in the case of the ATSP, if there are other time-related constraints, fusing them into the polynomial-size formulation may be easier. If a known full model is exponential in size (i.e., has exponentially many constraints and/or variables), the complexity of known separation algorithms for enumerating the constraints or known pricing algorithms for enumerating the variables will have an impact on the computation time.

One commonly asked question about an automatic modeller is, how does one know if the model produced is correct? The same question can be asked about a human modeller. With the latter, the common practice is to put the model into a solver, obtain an optimal solution, and present it to the end-user. If the end-user raises any concerns, then the modeller
addresses the concern by modifying the model. An automatic modeller can use a similar approach, and this would be subject to future studies.

3.3 Translation engine

As we can see from the ATSP example, there is a significant gap between a non-expert end-user’s initial description and the final MILP formulation. A translation engine will be required at different stages of automatic modelling. In the radiotherapy treatment planning example, we can see that from the problem description, some translation of the terms are required in order for the automatic modeller to recognise that the problem is equivalent to the sequencing of a set of elements and, therefore, has some similarities to an ATSP. This should prompt the automatic modeller to investigate further through HCI, and ultimately confirm that it is indeed an ATSP. Then, the translation of known COP features to known MILP constraints/models is required, e.g., in the case of the ATSP, the ACs and the SECs in the forms of the two sets of Set Partitioning Constraints and the MTZ SECs.

4 Conclusions

Automated mathematical modelling is an intricate, expert task that requires the development of several artificial intelligence components. In the first attempt to achieve this, we have developed an ontology for Mixed Integer Linear Programming (MILP) models. The ontology formulates expert mathematician knowledge in the domain in order to remove (or minimize) the need for the expert in the modelling of MILP-related problems. We showed that the automated formulation of straightforward MILP problems, such as the machine scheduling example, can be achieved using the ontology; however, modelling of more complex MILP problems like the Asymmetric Traveler Salesman Problem (ATSP) necessitates even a more comprehensive framework to overcome challenges such as the extraction of necessary complementary information about complex decision variables as well as the selection of best-suited model in presence of multiple possible models.

Our proposed framework to address the above challenges includes a problem elicitation module to find more complete problem specifications as well as a modelling module that formulates the formal mathematical model. A translation engine connects the non-expert problem domain to the expert mathematical model domain while a model evaluation engine evaluates multiple models for their accuracy and relevance.

We are currently planning for the development of other necessary components of our proposed framework utilizing different types of artificial intelligence. While the repository engines will be mostly based on symbolic rule-based systems, the translation engine and HCI components are planned to be developed using computational (learning) approaches.

References

[1] A. Astafiev, R. Prokofyev, C. Guéret, A. Boyarsky, and O. Ruchayskiy. Sciencewise: A web-based interactive semantic platform for paper annotation and ontology editing. In E. Simperl, B. Norton, D. Mladenic, E. Della Valle, I. Fundulaki, A. Passant, and R. Troncy, editors, *The Semantic Web: ESWC 2012 Satellite Events*, pages 392–396, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[2] C. Bessiere, F. Koriche, N. Lazaar, and B. O’Sullivan. Constraint acquisition. *Artificial Intelligence*, 244:315 – 342, 2017. Combining Constraint Solving with Mining and Learning.

[3] C. Bessiere, L. D. Raedt, L. Kotthoff, S. Nijssen, and B. O’Sullivan. *Data Mining and Constraint Programming: Foundations of a Cross-Disciplinary Approach*. Springer International Publishing, 1 edition, 2016.

[4] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[5] S. Buswell, O. Caprotti, D. Carlisle, M. Dewar, M. Gaëtano, M. Kohlhase, J. Davenport, P. Ion, and T. Wiesing. The openmath standard, 2019.

[6] D. Carlisle, R. Miner, and P. Ion. Mathematical Markup Language (MathML) Version 3.0, 2009.

[7] S. Ceschia, N. Dang, P. De Causmaecker, S. Haspeslagh, and A. Schaerf. The second international nurse rostering competition. *Annals of Operations Research*, 274, 03 2019.

[8] D. Chen, R. Batson, and Y. Dang. *Applied Integer Programming: Modeling and Solution*. Wiley, 2010.

[9] B. V. Dobrov and N. V. Loukachevitch. Development of linguistic ontology on natural sciences and technology. In N. Calzolari, K. Choukri, A. Gargemi, B. Maegaard, J. Mariani, J. Odijk, and D. Tapias, editors, *LREC*, pages 1077–1082. European Language Resources Association (ELRA), 2006.
[10] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 2962–2970. Curran Associates, Inc., 2015.

[11] M. Kohlhase. OMDoc – An open markup format for mathematical documents [Version 1.2]. Number 4180 in Lecture Notes in Artificial Intelligence. Springer Verlag, Aug. 2006.

[12] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown. Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA, pages 81–95. Springer International Publishing, Cham, 2019.

[13] M. Kumar, S. Teso, P. De Causmaecker, and L. De Raedt. Automating personnel rostering by learning constraints using tensors. 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), Nov 2019.

[14] S. Lohmann, S. Negru, F. Haag, and T. Ertl. Visualizing ontologies with VOWL. Semantic Web, 7(4):399–419, 2016.

[15] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. Journal of the ACM, 7(4):326–329, Oct. 1960.

[16] M. A. Musen. The protégé project: a look back and a look forward. AI Matters, 1(4):4–12, 2015.

[17] O. A. Nevzorova, N. Zhiltsov, A. Kirillovich, and E. Lipachev. Ontomathproontology: A linked data hub for mathematics. In P. Klinov and D. Mouromtsev, editors, Knowledge Engineering and the Semantic Web, pages 105–119, Cham, 2014. Springer International Publishing.

[18] T. P. Pawlak and K. Krawiec. Automatic synthesis of constraints from examples using mixed integer linear programming. European Journal of Operational Research, 261(3):1141–1157, 9 2017.

[19] V. D. Solovyev and N. Zhiltsov. Logical structure analysis of scientific publications in mathematics. In R. Akerkar, editor, WIMS, page 21. ACM, 2011.

[20] C. Steinruecken, E. Smith, D. Janz, J. Lloyd, and Z. Ghahramani. The Automatic Statistician, pages 161–173. Springer International Publishing, Cham, 2019.

[21] R. Thomas. Millenium mathematics project – Bringing mathematics to life. MSOR Connections, 4(3), 2004.

[22] D. Zhang, L. Wang, L. Zhang, B. T. Dai, and H. T. Shen. The gap of semantic parsing: A survey on automatic math word problem solvers. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1–1, 2019.