Intelligent ALMM System - an ontology approach for its Knowledge Base component

Abstract

This paper presents the implementation of a knowledge base supporting an intelligent system to solve problems of optimization especially problems of discrete production processes optimization called Intelligent Algebraic-Logical Meta-Model (ALMM) Solver. Using a unified description of selected optimization problems, an ontological knowledge base was designed, which allows for selective selection of Intelligent ALMM Solver components necessary to solve and model problems. Using the definitions of the properties of optimization problems, scalable components describing exemplary optimization jobs were selected. Ontology for this area was developed, with particular emphasis on the requirements of the ALMM Solver. Using the possibility of interactive communication with the ALMM ontology in the form of SQL queries in the experimental part of the work, exemplary queries for the designed Knowledge Base (KB) module were presented, and the response generated by the system is a scenario of intelligent selection of a set of components modeling and solving a given problem. Such an innovative approach allows for dynamic construction of algorithms solving problems of discrete optimization. The use of knowledge about the properties of the considered processes and ALMM technology universalizes the proposed KB system making it an intelligent and efficient tool for solving discrete optimization jobs. The key advantage of the proposed ontological approach is the ability to flexibly expand it and extend its use to other classes of problems which have already been described in the ALMM technology.

Introduction

As it has been already known, there are many discreet optimization problems that need to be solved in many practical areas such as production planning and scheduling, logistics, project management and much more. Many methods and algorithms of solving these problems using different strategies of their modeling have been developed. A number of solvers (so-called software tools) have been designed to help solve this type of discrete problems (Dudek-Dyduch and Korzonek 2016). The most popular are those using mathematical programming (MP) (Rostami, Creemers, and Leus 2019), especially linear discrete programming (e.g. LINGO environment), Constraint Logical Programming (CLP) (e.g. ECLIPSE environment), Evolutionary Programming or Agent Based Approach (Gedik et al. 2018), (Zhang, Ma, and Li 2015), (Liu et al. 2018), (Binh, Thanh, and Thang 2019), (Anselma et al. 2016), (Abdelhedi et al. 2016), (Kucharska, Grobler-Dbska, and Rczka 2017). Each of these approaches has its own advantages and limitations (Raczka et al. 2015). For example in case of use mathematical programming models there is no possibility to model especially logical dependencies (predicates), exclusions, If...Then constructions, etc. On the other hand, CLP method is characterized by poor efficiency of solving problems which has its own limitations caused by the existence of many decision variables, and in consequence this CLP method does not work for optimization (however, it works well in the search for acceptable solutions). Hence the last, new research is going in two basic directions. The first is to develop optimization tools using mixed modeling and optimization methods, mainly MP and CLP (Faris et al. 2019), (Piros et al. 2019), (Hao et al. 2019), (Sadik and Urban 2017), (Terenziani 2016). An example of such tools is the multi-job IBM system - ILOG CPLEX Optimization or Gurobis.

The second research direction includes new formal modeling methods and development of solvers (systems) of other types. This article is related just to second direction. In order to be able to design such a system it is necessary to create a description of the existing theory of discrete optimization in terms of ALMM paradigms and eventually exploit an Intelligent ALMM System architecture containing a modular knowledge base library.

The aim of this paper is to present the results of research concerning the latter stage, in particular:

- proposing an ontological approach to the presentation of knowledge about discrete optimization problems and methods of solving them;
- proposing the use of the Web Ontology Language (OWL) and Protégé environment for the organization and implementation of the KB;
- presenting selected stages of designing the ALMM ontology supporting the Intelligent ALMM System for an

exemplary area of job scheduling problems on a single machine;

- developing a potential scenario for solving exemplary optimization task using model components and DL or SPARQL queries.

The following chapters present the modeling paradigm and the concept of ALMM technology. The models of a several classes of discrete optimization problems, the so-called Algebraic Logical models (AL models) and the dependencies between them were presented. Then a technique of creating software representation of AL models using the component technology and designing the essential properties of particular classes of problems defined from the point of view of solving algorithms was proposed. The main contribution to this paper consists of the proposal of an ontological KB for the Intelligent ALMM system which stands the repository of the knowledge about solving a selected class of problems.

ALMM technology

Algebraic-Logical Meta-Model

The Algebraic-Logical Meta-Model is a general paradigm of creating formal models of problems of discrete optimization (Pandit et al. 2018), (Ezugwu 2019). It is different from the CLP and MP paradigm. It uses the notion of a multistage decision-making process, but formally presented in such a way that it is suitable not only for a single instance of the problem, but also for the whole set of instances of the given problem. The model of multistage dynamic decision process (MDDP) takes into account the fact that all restrictions may be time-dependent, while for common process (cMDP) these constraints are static. Let us recall the main concept of the Algebraic Logical Meta Model of the multistage dynamic decision process. The concept of the so-called 'generalized state' is used here, defined as a pair: a state and a time instant. A unified ALMM notation of selected problems is presented below, which will be considered in the simulation part of the paper and will be used to design an ontological knowledge base.

Definition 1

"The multistage dynamic decision process is a process that is specified by the sextuple $MDDP = (U, S, s_0, f, S_N, S_G)$ where U is a set of decisions, $S = X \times T$ is a set of generalized states, while X is a set of proper states, $T \subset R^+ \cup \{0\}$ is a subset of non-negative real numbers representing the time instants, the function $f : U \times S \to S$ is a partial function called a transition function, (it does not have to be determined for all elements of the set $U \times S$), $s_0 = (x_0, t_0)$, $S_N \subset S$, $S_G \subset S$, are respectively: an initial generalized state, a set of not admissible generalized states, and a set of generalized goal states, i.e. the states in which we want the process to take place at the end. Subsets S_G and S_N are disjoint i.e. $S_G \cap S_N = \varnothing$.

The transition function f is defined by means of two functions, $f = (f_x, f_t)$ where $f_x : U \times X \times T \to X$ determines the next state, $f_t : U \times X \times T \to T$ determines the next time instant. It is assumed that the difference $\Delta t = f_t(u, x, t) - t$ has a value that is both finite and positive. The assumptions that the transfer function is a partial one allows for various restrictions on "reasonable" decisions in different states to be taken into account by means of possible decisions $U_p(S)$ set defined as: $U_p(S) = \{u \in U : (u, s) \in Domf\}$ " (Dudek-Dyduch and Kucharska 2011), (Raczka et al. 2015), (Korzonek and Dudek-Dyduch 2017), (Dudek-Dyduch et al. 2018).

An important feature of this modeling paradigm is that decisions can be in general complex decisions. Therefore in the most universal case, sets U and X may be presented as a Cartesian product $U = U^1 \times U^2 \times \cdots \times U^m$, $X = X^1 \times X^2 \times \cdots \times X^n$, i.e. a complex decision $u = \left(u^1, u^2, \ldots, u^m\right)$, and a proper state $x = \left(x^1, x^2, ..., x^n\right)$. Elements u^i, $i = 1, 2, \ldots, m$ represent separate decisions that must or may be taken at the same time (or at the same stage in case of cMDP) and relate to particular objects. We can obtain the definition of the common multistage decision process (cMDP) from the definition of MDDP by reducing the set S to the set X and the transfer function f to the function f_x. Because all the elements specifying MDDP, i.e. U, S, s_0, f, S_N, S_G, $U_p(s)$ can be defined using both the algebraic and logical relations, hence the hybrid name "algebraic-logical". Based on the meta model recalled herein, AL models can be created for an extensive class of problems (and their instances) for which the solution may be presented as a sequence (or set) of decisions. The problems denoted as Π, consists of searching for optimal or admissible solution only. In case of admissible solution (admissibility problem) AL model is equivalent to a suitable multistage decision process, hence it is denoted as process $\boldsymbol{P}$, i.e. $\Pi = \boldsymbol{P}$. An optimization problem then is denoted as $\Pi = (\boldsymbol{P}, \boldsymbol{Q})$, where $\boldsymbol{Q}$ is a problem's criterion. An optimization or admissibility task (are instance of the problem) is denoted as (P, Q) or P respectively, where P is named an individual process and represents all limitations of the instance. One can use notation $P \in \boldsymbol{P}$.

ALMM provides a structured way of recording knowledge of the goal and all relevant constraints that exist within the problems modeled. In line with this idea, all information can be split into predefined basic components (elements) with appropriate links, defined both through algebraic and logical formulas. All individual processes within considered problem, $P \in \Pi$ are characterized by the same structure of decisions and the states respectively as well as definition of transfer function, nonadmissible states and goal states. For a set of decisions U, the important property of proces is the decision structure, regarding it is a complex or a simple decision, and also whether the coordinates take numerical or symbolic values (names). A symbolic value may be associated with a range of numbers, functions and relations determining the way of calculating the transfer function. In a simple case, the value of a non-numerical coordinate may be the name of the element belonging to the set that is given in a problem. Then we can say that the coordinate is an individual variable. A special and simplest case is a process in which a decision has only one coordinate, which is an individual variable. This process is called one-dimensional, un-

like the others which will be called multi-dimensional. There are no limitations imposed on types of the sets; in particular they do not have to be numerical ones. Thus values of particular co-ordinates of a state or a decision may be names of elements as well as some objects (e.g. finite set of symbols or sequence of actions etc.). Similarly as in the case of the set of decisions, in the set of states one can distinguish numeric and non-numeric coordinates. Within the non-numeric coordinates one should distinguish individual coordinates and coordinates of higher orders. The maximum order of state coordinates is one of the most important features of the process. Referring to state coordinates, it should be noted that individual coordinates have different roles in the model: they are used to check the conditions defining sets of goal states or nonadmissible states, to determine the quantities occurring in the definition of sets U_p and to calculate the values of other state coordinates or criterion values. As it is known (Dudek-Dyduch and Korzonek 2016) the properties of the transfer function include the elements described through the consecutive list of steps:

- checking whether the decision belongs to a set $U_p(s)$ of decisions possible in the state s,
- calculation of the function $\Delta t = f_t(u, x, t) - t$,
- calculation of the function: $f_x : U \times X \times T \to X$.

The characteristics of the method of calculating the transfer function encompasses the properties of the algorithm determining f_x, f_t (by default including Δt) in particular the stationarity of both functions and the properties and method of determining $U_p(s)$.

Considering the properties of the S_N set, the following cases can be distinguished: lack of limitations concerning set of states $S_N = \varnothing$, limitations concerning numerical coordinates (time, raw materials) and expressed by symbolic variables (e.g. for a travelling salesman's job) $S_N \neq \varnothing$.

For any problem Π, the S_G set cannot be defined as an empty set. The conditions defining it are related to a fixed final time (only for a coordinate t) or a fixed set of goal states (only for coordinates of x).

AL models and Software Representation models

AL models are most conveniently defined using the set theory language. As we know, this notation is equivalent to the combinations of logical formulas. An AL model gives the specification which stands the basis for the so-called Software Representation Models (SR Models). Let's denote the ordinary scheduling problem by the $\alpha|\beta|\gamma$ notation (Shen and Zhu 2018), (Meng et al. 2019), where α represents information on machines, β - restrictions referring to the jobs and γ a determined quality criterion. Below we consider the examples of AL models that will be used to build their software representation.

Example 1. A single machine scheduling problem without time restrictions $1|-|C_{max}$

A finite set of jobs J is given; the job names are natural numbers $J = \{1, 2, \ldots, n\}$, j - number of jobs. The function specifying the processing times of the jobs τ_j is given: $J \to \Re^+$ and function $w : J \to \Re^+$, assigning weights to the individual jobs. The sequence of jobs to be carried out on one machine must be specified so that the weighted average of the completion times is minimal. The process state is determined by a set of jobs performed (x should be understood as the name of the state and the set being its value at the same time), and the decision consists of determining witch job have to be performed in the next stage of the process. The value of the decision will be the name (number) of a selected job. The formal model is defined as follows:

- the set of states X is a set of all sub-sets of jobs $X = 2^J$,
- the set of controls U is equal to the set of all jobs $U = J$,
- there are no distinguished nonadmissible states, therefore $S_N = \varnothing$,
- set of goal states $S_G = \{(x, t) : x = J\}$,
- initial state $s_0 = (x_0, t_0) = (\varnothing, 0)$,
- transfer function $f : U \times X \times T \to X \times T$ is defined as follows:
  - $U_p(x, t) = J \setminus x$, therefore $U_p(s_0) = J$
  - $f_x(u, x, t) = x \cup \{u\}$, $f_t(u, x, t) = t + \tau(u)$

The transfer function therefore has the following properties:

- the set of possible decisions depends only on the proper state
- the set of possible decisions in the state x_{i+1} can be calculated by simple modification of the previous set $U_p(x_{i+1}) = U_p(x_i) \setminus \{u_i\}$, where $\{u_i\}$ is a simple decision,
- the f_t function does not depend directly on the state x but only on the decision u.

From the definition of the set $U_p(x, t)$ and the fact that $S_N = \varnothing$ it follows that each sequence of decisions $\tilde{u}$ corresponding to the selected permutation of jobs is an admissible solution.

Example 2. Scheduling problem with due time restrictions $1|C_j \leqslant d_j|C_{max}$

Let us modify the problem presented in Example 1 by setting deadlines d_j for completing jobs. Let the function $d : J \to \Re^+$ specify the latest deadlines for the completion of the individual jobs. The sets of states and decisions X, U, the set of generalized goal states S_G and the initial state s_0 remain defined as before, so: $X = 2^J$, $U = J$, $S_G = \{(x, t) : x = J\}$. However, the definition of a set of nonadmissible states changes. It is now determined by a set of all pairs (x, t) in which the time is greater than or equal to the deadline for completing a job, and the job does not belong to a set of completed jobs x.

$$S_N = \{(x, t) : \exists j \in J, j \notin x, d(j) \leqslant t\} \tag{1}$$

The transfer function f is defined as before. Whether or not a state is admissible depends on time. In this example, not every permutation of jobs will be an admissible decisions.

Example 3. TSP problem expressed in the AL terminology

An undirected graph $G = (J, R)$ is given, where:

- J - is a set of the vertices of the graph, having the interpretation of a set of cities that the travelling salesman has to visit exactly once,
- $R \subset J \times J$ - the relation between the vertices of the graph defining the connections between the cities.

The function $\tau = J \times J \rightarrow \mathbb{R}^+$ assigns to each edge (i, j) a positive number, having the interpretation of the distance between cities (travel time); for pairs not belonging to the R relation, the distance is equal to ∞. In the model, instead of distance, we take the travel time assuming its proportionality to the distance. It is convenient to present the model of this job using a set of initial states, but in order to be in accordance with the accepted scheme of a single initial state, we will introduce an additional 0-job and extend the relationship R and the function τ as follows:

$$\forall j \in J\,(0, j) \in R, \forall j \in J\,\tau\,(0, j) = 0 \tag{2}$$

Let us define the elements specifying the process $\boldsymbol{P} = (U, S, s_0, f, S_N, S_G, U_p)$ and provide their interpretation. The proper state x encompasses information about the set of visited cities, the first visited city and city where the travelling salesman actually is. The decision consists in determining of the next city. Thus, the decision values are the names (numbers) of the chosen cities. Formally we can rewrite the set of proper states as $X = X^0 \times X^1 \times X^2$, a proper state becomes a *3*-touple $x = \left(x^0, x^1, x^2\right)$, where:

- $x^0 \subset 2^J$- a set of all the cities visited except the last one,
- x^1- the number of the city where the travelling salesman is usually the last visited city,
- x^2- the number of the city to which the travelling salesman must return (for a state other than the initial x_0), this is also the first city in the trip.

The initial state $s_0 = (x_0, t_0)$ where $x_0 = (\varnothing, 0, 0)$, $t_0 = 0$. The set of decisions $U = J \cup \{0\}$, while a simple decision u will mean choosing the next city. The set of possible decisions includes the cities that can be reached without affecting the limits of the problem. This set is not time dependent: $U_p\,(x) = \left\{j \in J \backslash \left(x^0 \cup \left\{x^1\right\}\right) : \left(x^1, j\right) \in R\right\}$. Regarding the initial value of i we can define the transfer function formulae in the following form:

- when $i = 0$, what denotes an "artificial city" or "artificial begin":

$$f\,(x) = \begin{cases} x^0_{i+1} = \varnothing,\ x^1_{i+1} = u_i,\ x^2_{i+1} = u_i \ for\ i = 0 \\ which\ is\ also\ equal\ to: \\ x^0_{i+1} = \varnothing,\ x^1_i = u_0,\ x^2_1 = u_0 \ for\ i = 0 \end{cases} \tag{3}$$

- when $i > 0$

$$\begin{aligned} f_x\,(u_i, x_i) = x^0_{i+1} = x^0_i \cup \left\{x^1_i\right\}, \\ x^1_{i+1} = u_i,\ x^2_{i+1} = x^2_i \end{aligned} \tag{4}$$

Thus regarding above the function f_x can be considered for two cases, when $i = 0$, and for $i > 0$. The decision u_i means the choice of the city, which means the chosen city to which the travelling salesman is to go, or in case of a production process interpretation, the choice of the next workpiece to be processed. The increase of time during the process is represented by the dependency:

$$\Delta t\,(u_i, x_i) = \tau\left(x^1_i, u_i\right) \tag{5}$$

where the coordinate x^1_i denotes the number of the city in which the travelling salesman is located, the number u_i means the city to which the travelling salesman will travel. The set of goal states can be defined as:

$$S_G = \{(x, t) : x \in X_G\} \tag{6}$$

where:

$$\begin{aligned} X_G = \{\ \left(x^0, x^1, x^2\right) : \left|x^0 \cup \left\{x^1\right\}\right| = |J| - 1 \\ and\ \left(x^1, x^2\right) \in R\ \} \end{aligned} \tag{7}$$

The set of non-admissible states we define as:

$$\begin{aligned} X_N = \{\ \left(x^0, x^1, x^2\right) : \left|x^0 \cup \left\{x^1\right\}\right| = |J| \\ and\ \left(x^1, x^2\right) \notin R\ \} \end{aligned} \tag{8}$$

The set of goal states S_G as well as set X_N do not depend on the time.

As we can see from the above examples, the construction of the AL model consists in giving the definition of its base elements: X, U, f, s_0, S_N, S_G, $U_p\,(s)$. For each base element of the AL problem's model, the appropriate class is defined (the base elements of any instance are represented by objects of these classes). The classes corresponding to the base elements of the a problem are the so-called base components of the SR model. The SR models of AL models are implemented in C# and the relationships between them stand the basis for the structure of a KB on problems and their properties.

Intelligent ALMM System

The concept of Intelligent ALMM system is presented in this chapter. What distinguishes it from the solvers mentioned in the introduction is the fact that the Intelligent ALMM System has a knowledge base that is used to:

- store problem models and their components, represented in the ALMM technology,
- support the creation of new problems models by using the set of components stored in the database,
- store specifications of methods and algorithms for solving discrete problems that are presented in the ALMM technology,
- store definitions of general properties of discrete optimization problems and information about the properties that are met by particular problems,

- store rules combining the properties of problems with the possibility of using selected methods and algorithms to solve problems.

This approach allow us to overcomes the lack of ability for dynamic and flexible selection of algorithm components to solve a given problem. Figure 1 shows the modified structure of such system. It consists of 3 main parts: Knowledge Base, Intelligent User Interface and Solving Module (SM).

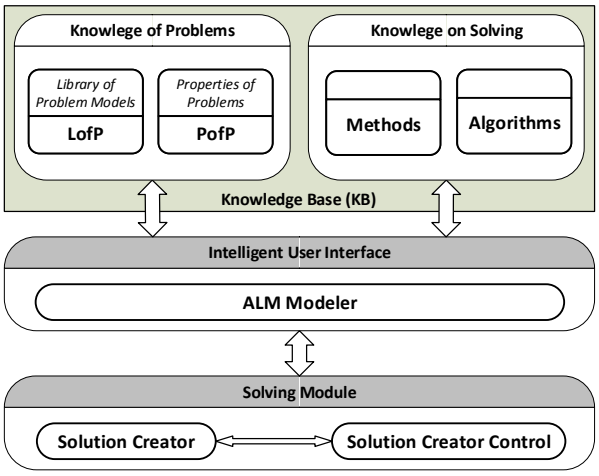


Figure 1: Modified structure of Intelligent ALMM System

The KB stores knowledge of problems and knowledge on solving methods and algorithms. The module Knowledge of Problems consists of 2 parts:

- the Library of Problems (LofP) stores the AL models and SR models of particular problems as well as contains a component repository for creating new models,
- the module Properties of Problems contain general definitions of the properties of processes and criteria as well as the relations between these properties and particular problems (components) contained in the Library of Problems.

The module Knowledge on Solving Methods and Algorithms stores specifications of methods and algorithms written in ALMM technology. It also stores the necessary rules for the selection of methods and algorithms for solving a problem. The user-selected method or algorithm controls through the Solution Creator Control module the calculations of the "executive" module Solution Creator. The Intelligent User Interface communicates with the user via the correspondingly provided menu.

The solution to the problem is represented as a sequence of decisions (usually the complex ones). Therefore, algorithms using AL models have to create the desired $\tilde{u}$ sequence and the corresponding trajectory in the state space.

At the initial stage, the user selects the problem model using ready-made models (model components) collected in the database or constructs his own problem model. Then he selects the solution method (algorithm). Listing 1 shows a possible access scenario to the SR or AL problem model stored in the KB or defined in terms of components on the basis of the ALMM ontology. The model is in the library when all of its SR components has been already created and stored in the library. The SR models are kept in the repository of problems.

Listing 1: The main part of the algorithm for user interface

```
1  Input: problem Π = (P, Q) or Π = P, Model
2  Output: solution a the given problem Π
3  // Π - problem, P - process, Q -
       criterion, LofPM - Library of
       Problem Models
4  ...
5  if (Π == P){
6       connect LofPMBase();
7            if (Model ⊂ LofPM) {
8                 switch (Model){
9                 case 'AL model':
10                     load ALModel();
11                     view(Model);
12                break;
13                case 'SR model':
14                     load SRModel(
                           ComponentsSRModel);
15                     Solve(Model);
16                break;
17                     default:
18                break;
19                }
20      else
21      {
22           if (ComponentsSRModel ⊂ LofPM){
23                MDDP = new SelectComponents
                       ();
24                Model = new SRModel(MDDP);
25                Solve(Model);
26           }
27           else {
28                MDDP = new SelectComponents
                       (SpecificProperties);
29                Model = new SRModel(MDDP);
30                addModel(LofPM);
31                Solve(Model);
32           }
33      }
34 }
35 else {//another optimization problem}
36 }
37 ...
```

The KB repository encompasses also a knowledge about the properties of problems. This repository of properties is used for two purposes:

- the use of component technology in the creation of AL models and SR models,
- assisting the user in the selection of methods and algorithms for solving selected classes of problems.

Ontological approach for the ALMM Knowledge Base

In order to overcome the problem of dynamic selection of problem solution components, the concept of the ontological ALMM KB approach has been proposed for construction the algorithm solving a given problem. In practice, there are ontologies with different degrees of formalization - from predefined vocabulary to knowledge models based on Descriptive Logic (DL) on which the OWL is based (Ashraf et al. 2015), (Munir and Anjum 2018), (Kuo-Wei Su 2016). The OWL is a language of knowledge representation that allows for the expression, exchange and processing of knowledge about a given problem area. Ontologies stored in the OWL syntax create classes, properties, individuals and data values (Qin and Zhu 2017), (Annane et al. 2018), (Kontchakov et al. 2014). In this work, the Protégé environment and the HermiT reasoner module were used for the ALMM ontology project for its KB repository. A new scenario generated by a DL Query or SPARQL query will stand the source of the names of the set of components, which are necessary to run the solver with the appropriate parameters while solving the selected problem.

Considered scheduling problems

In the experimental part of the paper an example of modeling semantic ontology composed of five problems of scheduling jobs on one machine (m=1) which where defined in Chapter 2, was considered. The purpose of the designed ontology is to generate answers to questions concerning the strategy of solving one of the above mentioned problems. In the $\alpha|\beta|\gamma$ notation these are the following problems $1||C_{\max}$, $1|C_j \leq d_j|C_{\max}$, $1|prec|C_{\max}$, $1|r_j|C_{\max}$, $1|C_{\max} \leq \tilde{C}|C_{\max}$. The AL models of the above problems are characterized by identical definitions of state and decision structures and the definition of the initial state of the process (Korzonek and Dudek-Dyduch 2017). The definitions of a set of prohibited states, a set of final states, the transfer function and a set of possible controls in a given state for a given problem $1||C_{\max}$ are assumed as default definitions. Some of the above definitions can be applied without any changes to the 4 other problems (see Figure 2).

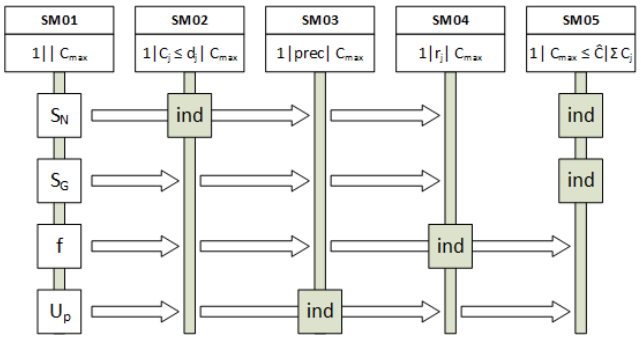


Figure 2: Selected problems and their ALMM components $\Pi = \{SM01, ..., SM05\}$

In the diagram presented here, white arrows represent the transfer of identical definitions of a given formulae from the SM01 problem to other formulae which are referred to as universal. Grey fields marked with the "ind" label (individual) refer to the specific definition of the problem and are referred to as private. For the SM02 problem, it is necessary to develop a specific definition of a set of prohibited states. For SM03, a new definition of a possible controls set in a given state is necessary. In case of the SM04 problem, the transfer function determining the next state requires redefinition. Only is case of SM05 problem it is necessary to change 2 definitions: a set of final states and set of nonadmissible states. It is therefore necessary to develop 3 common definitions to all problems, 4 default and 5 specific to each problem.

Assuming the above description of the knowledge about the problems being solved, the ALMM Construct Overview OWL was designed (see Figure 3). Taking into account the notation of the ALMM technology and Protégé environment, the diagram of relationships referring to five different problems of job prioritization was adopted. It represents all properties of particular problems and different definitions of target states sets, nonadmissible states, possible decisions and transfer functions respectively.

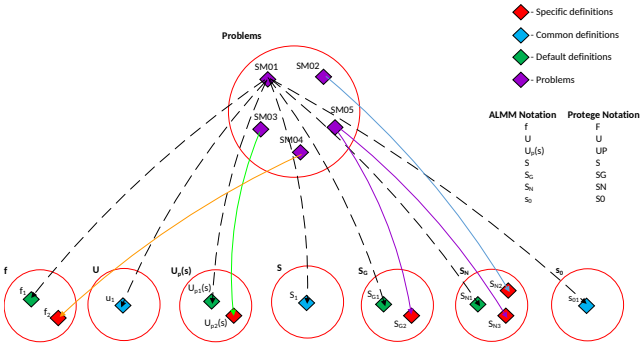


Figure 3: ALMM knowledge basis for an ontology model

Individual elements of the septuple describing particular problems stands the appropriate be classes, while instances of the SM01 - SM05 problems will be represented by individuals.

Ontology structure

The Problems class and its ALMMM Properties, describes the problem Π, the Criterion Properties and the Process Properties respectively. This relations between classes are symbolically marked with the relation "is-a" (see Figure 4).

The ALMM technology distinguishes between time-dependent (MDDP) and time-independent (cMDP) processes. In the KB ontology under consideration, the problems belong to the cMDP class representing a time-independent criterion, SingleMachine subclass. The subclass TDSingleMachine belongs to MDDP class, see. Figure 5.

The process properties are determined by the definitions of the basic AL elements which in Protégé will constitute individual components of the individual problems built into the

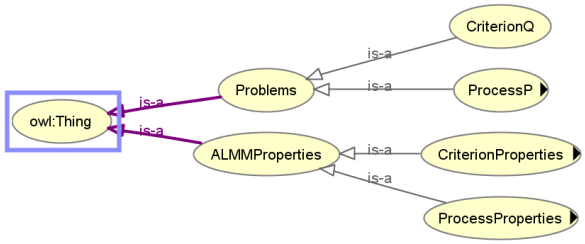


Figure 4: Basic class hierarchy in the ALMM ontology

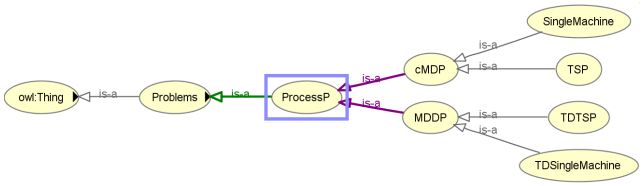


Figure 5: ALMM Ontology segments related to cMDP and MDDP problems

ALMM ontology. See the exemplary septuple in Figure 6 originated with F till U (regarding Protégé notation).

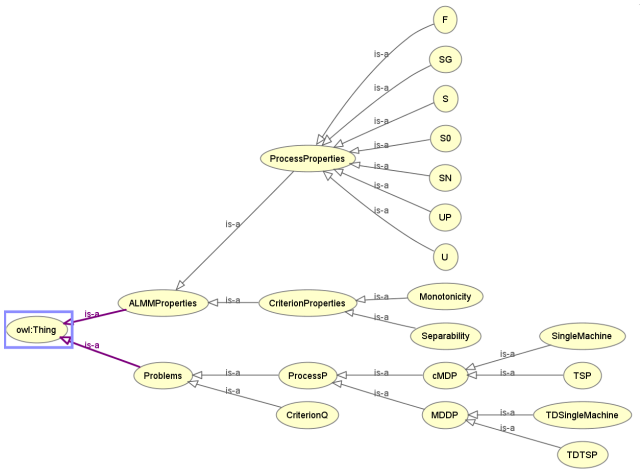
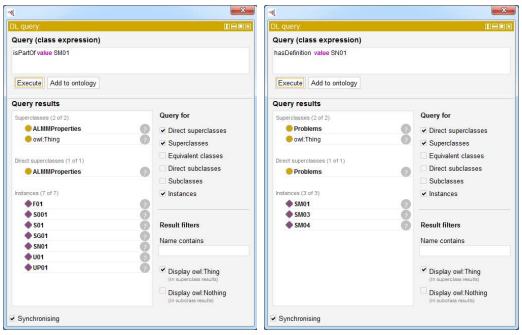


Figure 6: Hierarchy of classes with emphasis on the components septuple in scheduling problem

In such an architecture, particular optimization problems will be solved by selecting an already defined type of problem from the knowledge base or by isolating a septuple set or its subset which partially describes selected properties. The structure and properties of the optimization criteria are not considered in this paper and will be the next step in modeling of the KB for the Intelligent ALMM System.

Experiments

For the adopted ALMM KB Ontology architecture, examples of solution scenarios were generated, which will be the structural elements of the IT components of the model which solves selected problems. It has been assumed that user can create the query to the system and generated answer will control solving problems of a specific class. In such convention ontology supported ALMM system will be able to take into account the categories of problems or their properties. Figure 7 presents respectively the received set of elements describing the problem (a) and the problems described by its individual data (b).



(a) (b)

Figure 7: ALMM ontology response scenarios

In the numerical part of the project, Framework Jena Apache and dedicated API were used, allowing to operate KB data in the RDF (Resource Description Framework) format (Triple Store). The authors decided to choose this environment because it supports semantic W3C standards, such as SPARQL, RDF, OWL, providing an HTTP interface called Fuseki for RDF data type. In addition, Fuseki can be run as a standalone server to query and update the database.

In the adopted configuration of the Protégé environment, in the process of acquiring knowledge from the ALMM ontology, semantic query language SPARQL (Protocol And Rdf Query Language) was used, which allows to query structural and semi-structural data sources, explore data by querying unknown compounds, make complex connections from various databases in a single query, as well as transform RDF data from one dictionary to another. Using the configuration command for the server: `fuseki-server --update --mem /ds` we reserve an empty memory space for the data set. In this configuration, we can send the data set to the specified memory fragment, a `/ds` denotes the name of the file data which will be accessible via HTTP.

The implemented ALMM ontology named as `almm.owl` was saved on a dedicated server in the Laboratory of Applied Computer Sciences in the Center for Innovation and Transfer of Natural Sciences and Engineering Knowledge at the University of Rzeszów and made available at `www.neurosoft.edu.pl/ontologia/almm.owl`.

The query console module allowed for precise selection of appropriate elements of knowledge related to the problem being solved (see Listing 2). The example query presented above selects from the ALMM KB ontology all basic properties of the problem being solved in the context of generating the content of components set for the SR model. For the

$SM01$ problem, a group of properties describing the considered job was pointed: $F01$, $S001$, $S01$, $SG01$, $SN01$, $U01$, $UP01$.

Listing 2: The example query for the process properties

```
PREFIX almm: <http://www.neurosoft.edu.pl/ontologia/almm.owl#>
SELECT ?Properties
WHERE {alm:SM01 alm:hasDefinition ?Properties}
ORDER BY ?Properties
```

In the WHERE section, the RDF graph template is given, where the selected elements are replaced by variables starting with the question mark. The obtained results can be presented in two forms of text saving: Json format (see Listing 3) and CSV format (see Figure 8).

Listing 3: Result of query in Json format selecting the septuple for a single-machine problem

```
{
  "head":{
    "vars": ["Properties"]
  } ,
  "results": {
    "bindings":[
      {"Properties":{"type": "uri", "value": "http://www.neurosoft.edu.pl/ontologia/almm.owl#F01"}},
      {" ... #S001"}},
      {" ... #S01"}},
      {" ... #SG01"}},
      {" ... #SN01"}},
      {" ... #U01" }},
      {" ... #UP01"}}
    ]
  }
}
```

```
„Properties"
„http://www.neurosoft.edu.pl/ontologia/almm.owl#F01",
„http://www.neurosoft.edu.pl/ontologia/almm.owl#S001",
„http://www.neurosoft.edu.pl/ontologia/almm.owl#S01",
„http://www.neurosoft.edu.pl/ontologia/almm.owl#SG01",
„http://www.neurosoft.edu.pl/ontologia/almm.owl#SN01",
„http://www.neurosoft.edu.pl/ontologia/almm.owl#U01",
„http://www.neurosoft.edu.pl/ontologia/almm.owl#UP01",
```

Figure 8: Result of query in CSV format selecting the septuple for a single-machine problem

The resulting data about the component selection scenario allows further use of the file as a source for syntax analysis of the input data to determine a new formal grammatical structure for the SR model.
By further activity, the system user can select a specific problem from the presented list or complete the query to the system with another properties of the problem. It is a key feature of the system, because it enables dynamic selection of algorithm components before their compilation. Using the advantages of the Microsoft SQL data processing platform providing access to four servers in one system instance, the ALMM database of problems was designed in a multi-server version. In order to increase the efficiency of individual components, the implementation of the database of the library of problem models was carried out on a different server than the designed database of problem properties. This introduces comfort of security and independence of selected server failures affecting safety of the entire intelligent ALMM Solver. Proposing the use of MS SQL Server, and in particular, proposing separate operation on KB about problems and properties allows for parallel operation on both databases. This will facilitate distributed access for many customers in a later implementation.

Conclusions

This paper presents the implementation of an ontology knowledge based system supporting an intelligent ALMM Solver in order to solve problems of discrete production processes optimization. The system is based on the ALMM paradigm of algebraic-logical meta modeling and ALMM technology created on its basis. For exemplary optimization problems, an ontological knowledge base was designed, which allows for intelligent selection of system components necessary for modeling and solving problems. For this purpose, definitions of the properties of optimization problems and scalable components describing the considered issues were used. The ontology was implemeted using the Protégé environment taking into account the requirements of the ALMM Solver. The mechanism of interactive communication with the ALMM KB ontology in the form of SQL queries allows to potential user obtain a scenario of intelligent selection of a set of components which eventually solves a given problem. This is an innovative approach which is based on dynamic construction of algorithms for solving problems of discrete optimization. The designed KB ontology posses the knowledge about the properties of the considered processes described in terms of ALMM technology. This approach universalizes the resources of the proposed system making it an intelligent and efficient tool for solving discrete optimization tasks. The advantages of the proposed ontological approach include the possibility of flexible expansion and extension of its use to other classes of problems described in the ALMM technology. The disadvantage of the proposed approach is the risk of generating relationships by the Protégé reasoner which may not strictly comply with the specific requirements of the ALLM technology. Since the conducted experiments confirmed the possibility of using the ontological knowledge base on problems and their properties to optimize discrete processes, further research has been undertaken to expand the ALMM ontology with problems of job scheduling for parallel machines, with particular emphasis on different criteria for their optimization.

References

Abdelhedi, F.; Brahim, A. A.; Atigui, F.; and Zurfluh, G. 2016. Big data and knowledge management: How to implement conceptual models in nosql systems? In *8th International Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016)*, 235–240.

Annane, A.; Bellahsene, Z.; Azouaou, F.; and Jonquet, C. 2018. Building an effective and efficient background knowledge resource to enhance ontology matching. *Journal of Web Semantics* 51:51 – 68.

Anselma, L.; Piovesan, L.; Sattar, A.; Stantic, B.; and Terenziani, P. 2016. A comprehensive approach to now in temporal relational databases: Semantics and representation. *IEEE Transactions on Knowledge & Data Engineering* 28(10):2538–2551.

Ashraf, J.; Chang, E.; Hussain, O. K.; and Hussain, F. K. 2015. Ontology usage analysis in the ontology lifecycle: A state-of-the-art review. *Knowledge-Based Systems* 80:34–47.

Binh, H. T. T.; Thanh, P. D.; and Thang, T. B. 2019. New approach to solving the clustered shortest-path tree problem based on reducing the search space of evolutionary algorithm. *Knowledge-Based Systems*.

Dudek-Dyduch, E., and Korzonek, S. 2016. Almm solver for combinatorial and discrete optimization problems – idea of problem model library. In *Intelligent Information and Database Systems*, volume 9621, 459–469. Springer Berlin Heidelberg.

Dudek-Dyduch, E., and Kucharska, E. 2011. Learning method for co-operation. In *Computational Collective Intelligence. Technologies and Applications*, 290–300. Berlin, Heidelberg: Springer Berlin Heidelberg.

Dudek-Dyduch, E.; Gomolka, Z.; Twarog, B.; and Zeslawska, E. 2018. Intelligent almm system - implementation assumptions for its knowledge base. *ITM Web Conf.* 21:00002.

Ezugwu, A. E. 2019. Enhanced symbiotic organisms search algorithm for unrelated parallel machines manufacturing scheduling with setup times. *Knowledge-Based Systems* 172:15 – 32.

Faris, H.; Al-Zoubi, A. M.; Heidari, A. A.; Aljarah, I.; Mafarja, M.; Hassonah, M. A.; and Fujita, H. 2019. An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks. *Information Fusion* 48:67–83.

Gedik, R.; Kalathia, D.; Egilmez, G.; and Kirac, E. 2018. A constraint programming approach for solving unrelated parallel machine scheduling problem. *Computers & Industrial Engineering* 121:139–149.

Hao, P.; Zhang, G.; Martinez, L.; and Lu, J. 2019. Regularizing knowledge transfer in recommendation with tag-inferred correlation. *IEEE Transactions on Cybernetics* 49(1):83–96.

Kontchakov, R.; Rezk, M.; Rodríguez-Muro, M.; Xiao, G.; and Zakharyaschev, M. 2014. Answering sparql queries over databases under owl 2 ql entailment regime. In *The Semantic Web – ISWC 2014*, volume 8796, 552–567. Cham: Springer International Publishing.

Korzonek, S., and Dudek-Dyduch, E. 2017. Component library of problem models for almm solver. *Journal of Information and Telecommunication* 1(3):224–240.

Kucharska, E.; Grobler-Dbska, K.; and Rczka, K. 2017. Algebraic-logical meta-model based approach for scheduling manufacturing problem with defects removal. *Advances in Mechanical Engineering* 9(4):1–18.

Kuo-Wei Su, Chun-Hung Yang, P.-H. H. 2016. The construction of an ontology-based knowledge management model for departure procedures. *Advances in Aerospace Science and Technology* 1(1):37–426.

Liu, B.; Yao, L.; Ding, Z.; Xu, J.; and Wu, J. 2018. Combining ontology and reinforcement learning for zero-shot classification. *Knowledge-Based Systems* 144:42–50.

Meng, F.; Chu, D.; Li, K.; and Zhou, X. 2019. Multiple-class multidimensional knapsack optimisation problem and its solution approaches. *Knowledge-Based Systems* 166:1 – 17.

Munir, K., and Anjum, M. S. 2018. The use of ontologies for effective knowledge modelling and information retrieval. *Applied Computing and Informatics* 14(2):116 – 126.

Pandit, D.; Zhang, L.; Chattopadhyay, S.; Lim, C. P.; and Liu, C. 2018. A scattering and repulsive swarm intelligence algorithm for solving global optimization problems. *Knowledge-Based Systems* 156:12 – 42.

Piros, P.; Ferenci, T.; Fleiner, R.; Andrka, P.; Fujita, H.; Fz, L.; Kovcs, L.; and Jnosi, A. 2019. Comparing machine learning and regression models for mortality prediction based on the hungarian myocardial infarction registry. *Knowledge-Based Systems*.

Qin, H., and Zhu, L. 2017. Knowledge fusion and synchronization over ubiquitous ontology mapping. *Journal of Computer and Communications* 5(10):1–9.

Raczka, K.; Dudek-Dyduch, E.; Kucharska, E.; and Dutkiewicz, L. 2015. Almm solver: The idea and the architecture. In *Artificial Intelligence and Soft Computing*, 504–514. Cham: Springer International Publishing.

Rostami, S.; Creemers, S.; and Leus, R. 2019. Precedence theorems and dynamic programming for the single-machine weighted tardiness problem. *European Journal of Operational Research* 272(1):43 – 49.

Sadik, A. R., and Urban, B. 2017. Flow shop scheduling problem and solution in cooperative roboticscase-study: One cobot in cooperation with one worker. *Future Internet* 9(3).

Shen, J., and Zhu, K. 2018. An uncertain single machine scheduling problem with periodic maintenance. *Knowledge-Based Systems* 144:32 – 41.

Terenziani, P. 2016. Nearly periodic facts in temporal relational databases. *IEEE Transactions on Knowledge and Data Engineering* 28(10):2822–2826.

Zhang, F.; Ma, Z.; and Li, W. 2015. Storing owl ontologies in object-oriented databases. *Knowledge-Based Systems* 76:240–255.