ISSN 2007-9737

A Fuzzy Multi-Agent Problem: A General Depiction and its Logic Programming-based Application

Krystian Jobczyk, Patryk Gałczyński, Antoni Ligęza

University of Science and Technology,
Department of Applied Computer Science,
Poland

jobczyk@agh.edu.pl, galczynski.patryk@gmail.com

Abstract. A Multi-Agent Problem (MAP) may be seen as a class of planning and scheduling problems with multiple interacting intelligent agants. These problems may be naturally expressible and solveable by means of solvers of such programming languages as PROLOG. MAP is sometimes fuzzified because of a vagueness of information, for example – about abilities and preferences of interacting agents. Simoultaneously, MAP naturally stems from different variants of Constraints Satisfaction Problems (CSP). This property should be preserved in each fuzzy extension of this problem. According to these requirements and expectations a Fuzzy Multi-Agent Problem (FMAP) as referred to CSP and its PROLOG-based solutions are considered in the paper. Finally, a brief discussion of the achieved solutions is also carried out. An effectiveness of this PROLOG-based approach exploits a multi-valency-based approximation of fuzziness in programming contexts.

Keywords. Multi-agent problem, fuzzy multi-agent problem, logic programming, PROLOG, simple temporal problem under uncertainty.

1 Introduction

AMulti-Agent Problem (MAP) is usually identified with a class of different problems of in Artificial Intelligence (see: [28]) involved in a multiple of interacting and intelligent agents. Despite of some particular differences between problems from this class, at least one of the two following properties may be asssociated to them. It is A) an inventing of the action sequence in order to perform a goal or B) associating actions to agents that could be performed by them due to their skills.

Although a conceptual provenance of different problems under the banner MAP should be found in a class of different optimizations problems, such as: the so-called *Nurse Job Scheduling Problem* (NJSP)[1] – intensively explored in such works of Nottingham's school as: [3, 1, 5, 23, 21] – MAP remains close-related to Constraints Satisfaction Problems (CSP). One can venture to fomulate the thesis that MAP forms a unique correlate of a simplified temporal variant of CSP – the so-called *Simple Temporal Problem* (STP) – introduced and broadly discussed together with its extensions (STPU and STPP[2]) in such works of Dechter-Khatib's school as: [4, 15, 26, 18, 16, 29, 24, 11].

In practice, different combinations of MAP with CSPs have been elaborated under the banner of 'Multi-Agent Approaches to CSP' – as described in [19, 22, 20, 30]. These (usually sophisticated) synergic combinations have either a nature of asychronous solvers, such as: *Asynchronous Backtracking* (ABT) or *Asynchronous Weak-Commitment Search* (AWCS) – due to [30] or a character of distributed local search methods, such as: *Distributed Brekout Algorithm* (see: [30])

[1] This problem is also called *Nurse Rostering Problem*. NSP constitutes a classical optimization problem. Its objective is to determine the rotating shifts of the nursing shifts over a scheduled period (weekly or monthly)- see:[6].

[2] STPU forms an abbreviation from 'Simple Temporal Problem under Uncertainty' and STPP – from 'Simple Temporal Problem with Preferences'.

or *Environment, Reactive rules and Agents* (ERA) approach (see: [19]). It appears that MAP in the context of CST forces some new special requirements imposed both on agents activity and for their mutual interaction. For example, in ABT – each agent performs a distributed version of of the backtracking in asynchronous way and a total order among agents is required. In contrast – in ERA, agents move synchronously and cannot operate in the domain of other agents, etc. (see: [19].) Finally, an idea of the combination of MAP with CSP found its reflection in a synergic definition of *Distributed Constraint Satisfaction Problem* (DisCSP) as an algebraic foundation of the Multi-Agent CSP approach.

MAP sucessfully finds its programming-wise reflection in such languages of a declarative paradigm as PROLOG, Answer Set Programming (ASP) or – recently – in Bousi-PROLOG, which integrates fuzzy logic and logic programming for a use of dealing with approximate reasoning – due to ideas from [7, 8].

Meanwhile, an inherent nature of NJSP – as a conceptual foundation of MAS – sometimes required a piece of fuzziness because of a need to grasp vaguenees of information (for example, about the hospital objectives or personal preferences). Thus, a kind of a fuzzified MAP (FMAP) seems to be expected in order to provide a high quality scheduling tasks. A new step towards a fuzzification of NJSP involved in fuzzy goal programming (GP) models has recently been proposed in [2, 25].

Anyhow, it seems that fuzziness may be introduced in different manner to the optimization problems. In the context of MAS, it may be naturally introduced by different (sometimes unknown) degrees of ability of agents to perform their tasks. A similar comprehension of FMAP is just proposed in this paper.

1.1 The Paper Motivation

Independently of a visible success of the Multi-Agent Approaches to CSP and of a still increasing effectiveness of different specified algorithms, solvers and variants of dynamic programming, some difficultes of the proposed approches may be easily detected.

A Nurse Scheduling Problem – as a basis of Multi-Agent Problem should be rather seen a basis *reservoir* of possible, more specified formulations and it still waits for a more advanced fuzzy complementation. In particular:

- — Admittedly, Ernst's approach in [5] is mathematically general, but it refers to some simplified situations.
- — In contrast, Lepegue's approach from [17] is pretty detailed, but formalization of possible temporal constraints in it seems to be too excessive and it is counterintuitive.
- — Finally, a fuzzification from [25] more predicts some results than puts forward them in detail.

B It is not completely clear how to relate MAP to the simplified temporal CSP in terms of STP and its different extensions – as defined in [4, 15, 29, 24].

C It seems that a fuzzy and multi-valued logic-based approach to combining CSP with other 'entities' of fuzzy temporal-reasoning, such as preferences – due to [13, 12, 11, 10, 14] – does not indicate any appropriate method to grasp MAP in their contexts.

D It also seems – due to author's best knowledge – that no explicit approach to fuzziness has been proposed in the contaxt of MAP on the conceptual level.

E Finally, an expressive power of PROLOG and its solvers has not yet been completely explored with respect to any fuzzified version of MAP – independently of a variety of modern alternative tools (such as GP, ADOPT) used for a fuzzified NJP or Multi-Agent Constraints Satisfaction (see:[25, 19, 2].)

Meanwhile, the choice of the PROLOG-solvers in the context of FMAP requires a brief justification in the light of the fact that Bousi-PROLOG is

ISSN 2007-9737

just addressed to model vague knowledge and approximate reasoning. The solution is dictated by two facts. At first, development of Bousi-PROLOG still remains in progress, so it is relatively difficult to evaluate its expressive power. Secondly, exploting of PROLOG itself allows us to deeply recognize an expressive power and some adequacy of PROLOG for representation of FMAP and its workable subcases. Investigations of the paper partially stem from research presented in [9].

1.2 The Objectives of the Paper and its Organization

According to the requirements and needs, just formulated, and slightly against the tendency of the technology-oriented research on Multi-Agent Constraints Satisfaction – the paper forms a unique return to the conceptual foundation of the considered issues. In fact, this paper will be devoted to a new formulation of Multi-Agent Problem (MAP) and its fuzzy extension (FMAP). The newly introduced FMAP and DisCSP will be also referred to STPU as their conceptual correlates in order to detect some important similarities and discrepancies between them. Finally, some workable subcases of FMAP will be solved by means of PROLOG machinary. In addition, complexity of these solutions will be briefly discussed.

Novelty of the paper with respect to earlier approaches consists in:

N1 proposing a fuzzy modification of MAP – both in a practical and in a slightly genral depiction,

N2 describing both MAP and FMAP as a synergy syntesis of planning and scheduling components in a referrence to both STPU and DisCSP,

N3 detecting both similarities and discrepancies between FMAP and STPU and DisCSP.

N3 considering the PROLOG-solvers for some workable fuzzy cubcases of FMAP.

The rest of the paper is organized as follows.Section 2 introduces a terminological background of the paper analysis. Section 3 contains both the (more) practical depiction of MAP, its generalization and FMAP. Finally, a brief taxonomy of different constraints imposed on these problems is also put forward here. Section 4 is devoted to the programming-wise aspects of FMAP. Section 5 contains closing remarks.

2 Terminological Background

Before we move to the main body of the paper, we preface our analysis by introducing the concepts of STP and STPU as further extension of STP. The terminological framework determined by these concepts allows us to better grasp a conceptual specificity of both MAP and FMAP, introduced in detail in Section 3.

Simple Temporal Problem[3] forms such a simple subproblem of TCSP that all constraints are specified by a single interval. More formally, STP constitutes the kind of the Constraints Satisfaction Problem, where a constraint between time-points X_i and X_j is represented in the constraint graph as an edge $X_i \rightarrow X_j$, labeled by a single interval $[a, b]$ that represents the constraint:

$$a \leq (X_j - X_i) \leq b. \tag{1}$$

Solving STP – due to [24] – means to find an assignment of values to variables such that all temporal constraints are satisfied. It appears that STPs can be solved in polynomial time, whereas the complexity of a general TCSP belongs to a class of NP-problems[4].

In [29] STP was extended by considering contingent events, whose occurrence is somehow controlled by exogenous factors sometimes referred to 'Nature'. In this way we introduce *Simple Temporal Problem under Uncertainty* (STPU) as a further specification of STP, which still

[3] The conceptual basis of this chapter presentation makes use the following works [4, 15, 29, 24].

[4] STP may be also associated to the so-called distance edge-weighted graph $G = \langle V, E \rangle$ with V is a set of vertices and E – a set of edges between them, where each edge $i \rightarrow j$ (between vertices i, j) is labeled by a weight a, representing the linear constrain $(X_j - X_i) \leq a$

ISSN 2007-9737

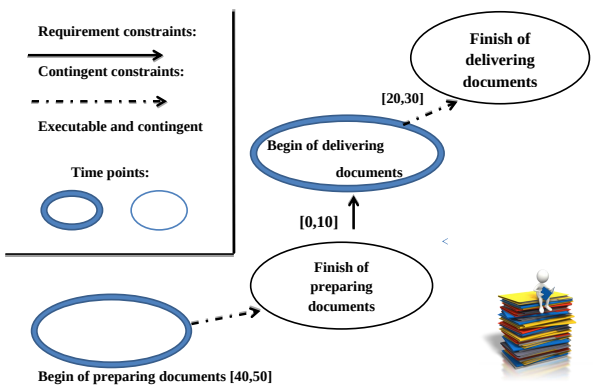

Fig. 1. An example of STPU

preserves some important point of the STP-based conceptualization (such as a controllability of start points of an agent activity).

Nevertheless, the end times (end time points) are divided into two classes: *required (requirements)* ('free' in [29]) and the *contingent* ones. It immediately introduces a distinction between STPU-variables for the *executable, required* and *contingent* time points. The first ones – are associated to performing agents, the second ones – are assigned by 'external' worlds (factors).

Definition 1. (**STPU**.) *A Simple Temporal Problem with Uncertainty* (STPU) is such a 4-tuple N = $\langle X_{exec}, X_{cont}, R_{eqr}, R_{cont}\rangle$ that:

$X_{exec} = \langle e_1 \ldots, e_k\rangle$: is the set of executable time-points,

$X_{cont} = \langle c_1 \ldots, c_k\rangle$: is the set of contingent time-points,

R_{req} - set of required constraints ,

R_{cont} - set of contingent requirements .

Example. *Consider the following scenario connected to the following two activities:* `preparing a documentation` *and* `delivering this documentation to the office`. *Assume that you can control their star points, but you cannot completely control their finishing. Assume also that you want to deliver a documentation relatively immediately after its preparing. Again, you can control a beginning of its delivering to the office, but you do not know where it will be finished. The contingent and required time points and possible constraints of this scenario are depicted in Fig.1.*

On a base of a general definition of CSP, the following definition of DisCSP has been elaborated in [30].

Definition 2. (**DisCSP** forms a 5-tuple $\langle A, X, D, C, \phi\rangle$, where $\langle X, D, C\rangle$ is CSP, A is a set of agents and $\phi : X \rightarrow A$ forms a function assigning variables from X to agents from A.

This generic formulation that combines a multi-agent problem with CSP may be specified towards some additional requirements imposed on agent interaction and capabilities.

3 Multi-Agent Problem and Fuzzy Multi-Agent Problem

In this chapter, our initial depiction of MAP will be put forward, but its formulation is motivated by Nurse Rostering Problem. This solution allows us to grasp many intuitions that find their reflection in a definition of FMAP and in the appplication-based part of the paper. In order to make it, let us note that each well-founded depiction of MAP must satisfy the following general criteria.

C1 A finite (non-empty) of agants schould be given,

C2 Agents should be involved in some activities in some time periods and subperiods (for examples: days and shifts),

C3 There are some hard constraints imposed on agant activities thatm must be absolutely satisfied to perform the task,

C4 There are some soft constraints imposed on agent activities that may be satisfied.

ISSN 2007-9737

Taking into account these general criteria, one can formulate the following generic Multi-Agent Problem as follows:

Multi-Agent Problem (MAP). Consider a factory with n-agents working in a rhythm of the day-night shifts: D–the day shift and N–the night shift. Generally – each day at least one person must work at the day shift and at least one – at the night one. Each agent has "working shifts" and "free shifts". These general rules of scheduling is constraints in the following way.

HC1 The charm of the shift organization should be fair: each agent must to have equally: 2 day-shifts and 2 night-shifts.

HC2 Each agent can be associated to at most one shift,

HC3 Some shifts are prohibited for agents,

HC4 Length of the shifts sequences associated to each agent is restricted,

HC5 Quantity of the shifts in a scheduling period is restricted,

HC6 Quantity of the shifts per a day is restricted.

The MAP consists in a construction of a scheduling diagram, which respects all these constraints.

The constraints HC1-HC6 form an exemplification of the so-called *hard constraints* as these constraints, which *should be satisfied* in a scheduling task. They ensure a feasibility of the scheduling task. *The soft constraints* may not be satisfied, but a degree of their satisfaction is a measure how good is a performed plan. In order to make the requirements with respect to the hard constraints more liberal, the so-called relaxation, i.e. a weakening of the strong constraints will be used. We often use this solution, when satisfaction of all hard constraints leads to an inconsistency.

A further relaxaction of requirements or expectations allows us to consider other entities, such as preferences or fuzzy requirements. Only the last ones will concern us is this paper. In general, the *fuzzy constraints* might be the entities of a nature, which introduces a kind of fuzzines to MAP.

The fuzziness may be determined by uncertainty, partial observability, a lack of a complete knowledge, etc. In this paper, a meaning of 'fuzziness' is restricted to an existence of different (usually unknown) degrees of agent ability to perform their tasks in the context of MAP. All these degrees determines a [0,1]-continuum with 0 as a complete lack of such a disposition and with 1 as a full disposition to perform the appropriate tasks.

In such a conceptual framework, *Fuzzy Multi-Agent Problem (FMAP)* may be viewed as *Multi-Agent Problem* equipped with additional fuzzy requirements imposed on the task performing. An exemplary, paradigmatic formulation of FMAP is given as follows.

Fuzzy Multi-Agent Problem (FMAP). Consider a factory with n-agents working in a rhythm of the day-night shifts: D–the day shift and N–the night shift. Generally – each day at least one person must work at the day shift and at least one – at the night one. Each agent has "working shifts" and "free shifts". These general rules of scheduling is constrained in the following way.

HC1 The charm of the shift organization should be fair: each agent must to have equally: 2 day-shifts and 2 night-shifts.

HC2 Each agent can be associated to at most one shift,

HC3 Some shifts are prohibited for agents,

HC4 Length of the shifts sequences associated to each agent is restricted,

HC5 Quantity of the shifts in a scheduling period is restricted,

HC6 Quantity of the shifts per a day is restricted.

Assuming also an agent $n_k \in N$ and the chosen (real) parameters m, M and α Different soft constraints and preferences of a general form are also considered in the scheduling procedure.

SC7 A quantity of shifts in a scheduling period is established,

SC8 A scheduling charm's covering by shifts in a scheduling period is established,

SC9 A length of the shifts sequence associated to an agent is fixed,

Fuzz1 A number of degrees of agent ability to perform actions is a natural number from the set $1, 2 \ldots, m$,

Fuzz2 An agent n_k prefers to perform an action a with a degree $\alpha \in 1, \ldots M$.

The FMAP consists in a construction of a scheduling diagram, which respects all these *hard* and *soft* constraints and the fuzzy requirements.

Although the list of possible fuzzy requirements might be enlarged, let us underline that both **Fuzz1** and **Fuzz2** rather constitute a *scheme* or a class name for different particular fuzzy constraints: **Fuzz2** – for the constraints introduced by preferences, but **Fuzz1** – for fuzzy constraints defined without them. Thus, both classes have a whiff of a paradigmatic generality.

3.1 Types of Constraints of FMAS and their Arithmetic Representation

Different approaches to a representation of both hard and soft constraints and preferences has been proposed. Majority of them grasps an arithmetic represnetation based on a calculus of characteristic functions – due: [17, 2, 25], but a multitude of different additional indexed parameters (c.a. 20) in their arithmetic depiction often makes an idea of representation only partially elusive.

Thus, a restricted set of parameters is introduced for a mathematical representation of temporal constraints imposed on (F)MAP. Instead of agent skills we will consider agent roles (contracts)[5]:

— $N = \{n_1, n_2 \ldots, n_k\}$ as set of agents (agents),

— $R = \{r_1, r_2, \ldots, r_k\}$ as a set of roles (contracts)

— $D = \{d_1, d_2, \ldots, d_k\}$ as a set of days in a week,

— $Z = \{z_1, z_2\}$ as a set of admissible shifts during days from *D*,

— $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ as a set of actions.

[5]All of these constraints are typical for scheduling problems of this type to be known as (usually) NP-hard – see: [3].

It enables of representing MAP by its formal instances in the form of the triple

$$(N, D, Z, A, HC, SC), \tag{2}$$

where N, D, Z are given as above and HC denotes a set of hard constraints imposed on actions from A and their performing. Similarly, FMAP may be given by the n-tuple of the form:

$$(N, D, Z, A, HC, Fuzz), \tag{3}$$

where N, D, Z and HC are given as above and SC and $Fuzz$ denote a set of soft constraints and fuzzy requirements (*resp.*) Introducing SC to the n-tuple (1) follows from the adopted hierarchy of constraints. The hard constraints cannot be violated, the soft ones may be violated, but they should be satisfied before fuzzy constraints.

This notation allows us to elaborate the following representation of hard and soft constraints. Since their list is not exhaustive[6], it might be relatively naturally extended. A formal representation of each constrain is based on a characteristic function $X_{n,d,z}$ defined as follows[7]:

$$X_{n,d,z} = \begin{cases} 1 & \text{if an agent } \textit{n} \text{ works a shift } \textit{z} \text{ in a day } \textit{d}, \\ 0 & \text{otherwise.} \end{cases}$$

HC 1: The charm of the shift organization should be fair: each agent must to have equally 2–day shifts and 2–night shifts. Assume that Z_{day} denotes a set of day-shifts and Z_{night} – a set of night-shifts. It enables the following formal representation:

$$\sum_{z \in Z_{day}} X_{n,d,z} = 2 \wedge \sum_{z \in Z_{night}} X_{n,d,z} = 2. \tag{4}$$

HC 2: Each agent can be associated to at most one shift.

$$\sum_{z \in Z} X_{n,d,z} = 1. \tag{5}$$

[6]This fact plays no important role as the main objective of this juxtaposition consists in the *quantitative representation* alone, which will be later combined with qualitative temporal constraints (of Allen's sort) for a use of further investigations.

[7]This binary representation can be also exchanged by a classical one: $X_{n,d} = z$ as presented in [3].

ISSN 2007-9737

HC 3: Some shifts are prohibited for an agent *n*. If Z_n denotes a set of shifts prohibited for a agent $n \in \{N_1, N_2 \ldots, N_k\}$, then this constraint may be mathematically depicted as follows:

$$\sum_{z \in Z_n} X_{n,d,z} = 0. \tag{6}$$

HC4: Length of the shifts sequence associated to an agent. This constraint defines a restriction for a sequence of shifts associated to an agent n. If we denote the minimal and the maximal number of shifts for an agent n by m and M (resp.), then HC4 may be depicted as follows:

$$m \leq \sum_{d}^{m_z+d} X_{n,d,z} \leq M\,. \tag{7}$$

HC 5: Quantity of shifts in a scheduling period. It defines the minimal and the maximal quantity of shifts during a given scheduling period (day) – associated to a single agent n.) If s is the minimal and S– the maximal quantity of shifts possibly associated to the agent in the scheduling period, then we get:

$$s \leq \sum_{d \in D} X_{n,d,z} \leq S\,, \tag{8}$$

where $X_{n,d,z}$ is defined as above.

Obviously, all the constraints listed above are not exhaustive and their list may be naturally enlarged. However, we interrupt their presentation in this point for a cost of a new presentation of FMAP in a more general depiction.

4 General Formulation of a Multi-Agent Problem

In this section a kind of a generalization of MAP and FMAP is put forward in order to provide a formal distinction criterion between MAP and FMAP. The conceptual framework of this generalization is determined by a set of agents, each of them possessing specific skills, a set of some temporal tasks to be completed and associated to agents.

Each agent can accept only tasks consistent with its skills[8]. Below, a generic, simple formalization of this problem is proposed.

Consider a set $\mathcal{A} = \{A_1, A_2, \ldots A_n\}$ of n agents. Each agent can possess one or more skills. Let $\mathcal{S} = \{S_1, S_2, \ldots S_k\}$ denote the set of predefined skills. Assume σ is the function defining a two-valued measure for all the skills of any agent; so σ is defined as:

$$\sigma\colon (\mathcal{A}, 2^{\mathcal{S}}) \mapsto \{0, 1\}.$$

For practical reasons, it is convenient to represent this function in a tabular (matrix) form as follows:

	S_1	S_2	$\ldots$	S_k,
A_1	$\sigma_{1,1}$	$\sigma_{1,2}$	$\ldots$	$\sigma_{1,k}$,
A_2	$\sigma_{2,1}$	$\sigma_{2,2}$	$\ldots$	$\sigma_{2,k}$,
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$,
A_n	$\sigma_{n,1}$	$\sigma_{n,2}$	$\ldots$	$\sigma_{n,k}$.

where $\sigma_{i,j} = \begin{cases} 1, & \text{if } S_j \in \sigma(A_i)^9, \\ 0, & \text{otherwise.} \end{cases}$

Similarly, consider a set $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ of tasks to be executed. Each tasks, in order to be executable by an agent, requires some specific skills. Assume θ is the function defining all the skills required to execute a specific task; so θ is defined as:

$$\theta\colon (\mathcal{T}, 2^{\mathcal{S}}) \mapsto \{0, 1\}.$$

Again, for practical reasons it is convenient to represent this function in a tabular (matrix) form as follows:

	S_1	S_2	$\ldots$	S_k,
T_1	$\theta_{1,1}$	$\theta_{1,2}$	$\ldots$	$\theta_{1,k}$,
T_2	$\theta_{2,1}$	$\theta_{2,2}$	$\ldots$	$\theta_{2,k}$,
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$,
T_m	$\theta_{m,1}$	$\theta_{m,2}$	$\ldots$	$\theta_{m,k}$.

[8] Further auxiliary constraints (e.g. Allen's type constraints for execution periods of certain actions) or extensions (e.g. parallel execution of actions by a single agent) are possible

where $\theta_{i,j}$=

$$\begin{cases} 1, & \text{if } S_j \in \theta(T_i)^{10}, \\ 0, & \text{otherwise.} \end{cases}$$

For simplicity, it is assumed that a single task can be executed by a single agent, one task at a time. Task T_j can be executed by agent A_i if and only if the agent possesses all the required skills. Formally, skills associated to tasks (obtained by the projection π on $2^{\mathcal{S}}$ in a domain of θ) should be contained in skills (obtained by the projection on $2^{\mathcal{S}}$ in a domain of σ) associated to agents from $\mathcal{A}$. Symbolically:

$$\pi_{2^{\mathcal{S}}}\Big(dom\{\theta(T_l, S_j)\}\Big) \subseteq \pi_{2^{\mathcal{S}}}\Big(dom\{\sigma(A_i, S_j)\}\Big),$$

and the execution can start whenever the agent is free; this holds for all $i \in \{1, 2, \ldots n\}$, $j \in \{1, 2, \ldots k\}$ and $l \in \{1, 2 \ldots, m\}$. Now, MAP consists in efficient assignment of all the tasks to given agents, so that the tasks can be executed, all the constraints are satisfied.

In this depiction, a distinction between MAP and FMAP reflects itself in another way of defining of functions σ and θ, which can take values from the whole interval [0,1]. Formally, each ρ_{ij} (for $i = 1, \ldots, k$, $j = 1, 2, \ldots, n$) from the definiting table:

	S_1	S_2	$\ldots$	S_k,
A_1	$\sigma_{1,1}$	$\sigma_{1,2}$	$\ldots$	$\sigma_{1,k}$,
A_2	$\sigma_{2,1}$	$\sigma_{2,2}$	$\ldots$	$\sigma_{2,k}$,
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
A_n	$\sigma_{n,1}$	$\sigma_{n,2}$	$\ldots$	$\sigma_{n,k}$

satisfies the condition: $\rho_{ij} \in [0, 1]$[11].

5 FMAP *versus* STPU *versus* DisCSP

Fuzzy Multi-Agent Problem has already been introduced not only in a practical NJSP-based depiction, but also – in a more general formulation. This two-side presentation allows us to propose an outline of a comparative analysis between FMAP,STPU and DisCSP.

[11]Necessarily, the two-valued 'membership' $\in$-relation is exchanged for a fuzzy 'membership'-relation in the condition $S_j \in \sigma(A_i)$.

5.1 FMAP versus STPU

It is not difficult to detect both a couple of similarities between FMAP and STPU, and some discrepancies. On one hand, constraint satisfaction plays an important role in both classes of problems. In addition, a similar taxonomy of constraints – divided (at least) into two groups: hand and soft constraints – is usualy considered in both cases. Nevertheless, the role of the constraints and their satisfaction seem to be a little bit different. In fact, the problem forms a consitutive components of STPU, its real conceptual 'core'.

This conviction may be easily verified in the light of STPU-definition as an extension of STP that forms a simplified TCSP – due to [29] repeated in Section 2. Meanwhile, different types of constraints constitute *only* an *additional component* of FMAP – as imposed on performing the appropriate scheduling tasks. Conversely – agents, their mutual iteraction and their capabilities constitute FMAP as such a one, whereas these componants may be radically restricted or even omitted in the generic formulation of STPU.

Some further observable discrepancy between STPU and FMAP stems from a semantic discrepancy between *fuzziness* and *uncertainty* as their founding concepts. Although fuzziness (in particular – in a sense given by fuzzy logic) may be exploited to represent and model uncertainty, the fuzziness seems to rather refer to a vague, imprecise nature of different modeled entities or states(such as as: 'small' disposition of an agent to work). In contrast – uncertainty seems to rather refer to some epistemic unclarity in acting and reasoning (such as: partial observability, a lack of some premises in reasoning). This fact may be rendered in alternative way as follows: uncertainty refers more to *epistemic* capabilities or skills of agents or an external observer, meanwhile fuzziness grasps more an *ontological* side of the systems/problems.

Finally, some interesting aspects of STPU and FMAP are ellucidated by the category theory-based concepts of graphs and monoids. MAS may be viewed as the organizing thoughs about agants acting on objects.

This property due to [27], p. 115 – may be algebraically rendered by monoids. In contrast, STPU materializes an idea of information flow (between nodes), that – due to Spivak's conviction from [27], p. 115 – may be naturally represented by systems of nodes and arrows (vertices). It explained why STPU is naturally associated to graphs and different methods of the graph search may find an application in STPU-solutions.

5.2 FMAP versus DisCSP

Let us recall that FMAP is defined as the tuple $(N, D, Z, A, HC, SC, Fuzz)$, where where N, D, Z and HC are the set of agents, D and Z are sets of days and shifts (resp.) and SC and $Fuzz$ denotes a set of soft constraints and fuzzy requirements (*resp.*) In other words, FMAP may be viewed as a triple (N, CSP, A) – as both HC, SC and SC may be commonly considered as CSP-components. Meanwhile, DisCSP that may be identified with the tuple: $\langle N, CSP, \phi\rangle$, where N is as earlier is CSP and ϕ forms a function assigning variables from CSP to agents from N. This new depiction of both FMAP and DisCSP explaines a conceptual referrences between them: DisCSP may be seen as a (slightly modified) reduct of FMAP. Conversely, FMAP may be seen as DisCSP equipped with an explicitly given set of actions to be associated to agents during the scheduling task. The role of this component allows us to see DisCSP rather in a perspective of scheduling task, meanwhile FMAP may be also seen in a perspctive of planning.

6 Programming-wise Aspects of Fuzzy Multi-Agent Problem

In this section, we intend to face the *Multi-Agent Problem* in order to illustrate how to solve some of its workable subcases by means of SWI-PROLOG-solvers. Earlier arrangements from Section 3 and Section 4 allow us to formulate a couple of general guidelines of the solver construction. In particular, the proposed representation method for hard and soft constraints in Section 3 suggests a choice of their arithmetic, summeric depiction in PROLOG.

Meanwhile, the proposed general depiction of FMAP from Section 4 suggests to associate agents to their skills or degrees of their abilities (to work).

6.1 A More Detailed Specification of Requirements

In order to illustrate this method of the solver construction, let us assume that a non-empty set $N = \{X, Y \ldots\}$ of agents and a non-empty set $D = \{1, 2, 3, 4, 5\}$ of working days (for simplicity we omit shifts during a day) are given.

In such a framework, the PROLOG-solver task is to give a schedule respecting the temporal constrains imposed on a task performing and the corresponding agent activity. The achieved solutions will be represented by lists of the form:

$$X = [X1, X2, X3, \ldots, Xk], \tag{9}$$

$$Y = [Y1, Y2, Y3, \ldots, Yl], \tag{10}$$

where $X(i), Y(j)$ are characteristic functions representing activity of agents *X* and *Y* during *i*-day and *j*-day (*resp.*) for $k, l \in \{1, 2 \ldots, 7\}$.

In general, two following types of situations may be considered:

crisp-type situations, when $X(i)$ and $Y(j)$ take only two values: 0 or 1 for $i, j \in \{1, 2, \ldots, 7\}$ or

fuzzy[12]-type situations, when $X(i)$ and $Y(j)$ take more than two values for $i, j \in \{1, 2, \ldots, 7\}$, for example: 0,1,2,3,4. We adopt natural numbers because of restrictions of PROLOG-syntax, which is not capable of representing values from $[0, 1]$. Nevertheless, we intend to think about these values as about normalized values. Namely, we will interpret 1 – taken from a sequence $1, 2 \ldots, k$ – as $\frac{1}{k}$, 2 as $\frac{2}{k}$, *k* as $\frac{k}{k} = 1$, etc.

[12] More precisely: multi-valued situations.

6.2 General Specification of the PROLOG Code

SWI-PROLOG solvers will be exploited to find solutions for scheduling tasks of FMAP. As mentioned above, each admissible solution will have a form of the appropriate list of natural numbers.

The required constraints imposed on the given scheduling tasks will define the appropriate 2- argument predicate `schedule(X,Y)`, where variables X, Y represents a pair of operating agents. The propoer body of the predicate definition will contain the following components:

1. *a general specification of each agent* as a list of its working days in a scheduling period, for example, `X = [X1, X2, X3, X4]`, `Y = [Y1, Y2, ..., Y10]`, etc.
2. *an association the admissible degrees of ability* to agents – introduced by the code of the form: `X ins 0...1`, *X ins 0,..., 5*, etc.
3. *an arithmetic restrictions on agent during a scheduling period*, for example, `sum(X, #<, 3)`,
4. *a logical restriction on the agents interaction*, for example: $X1\#/Y1, Y1\#/Y1$ (the first working day of X is not a working day for Y and conversely).

The expected solutions in a form of lists (9)-(10) have just been specified.

6.3 Crisp Cases

We will consider 3 subcases of MAP dependently on the problem complexity. In all the cases: 1 – in solution-lists – represents a state when an agent n_i occurs in a work and 0 – otherwise.

As the first one, a case of 2 agents working 5 days in a week (with a week restriction for each of them equal to 4) will be considered. It will be shortly denoted as MAP(2, 5, 4). It is assumed that a maximal sum of working days of each of them does not exceed 5.

Case 1 – MAP(2, 5, 4). In this case, we consider the following situation:

— 2 agents operating during a 5-day scheduling period,

— only two degrees of ability to work are associated to the agents,

— a restriction imposed on number of activities (shifts) is smaller than 4,

— each day (1-5), a working activity of the first agent excludes a working activity of the second agent and conversaly.

The requirements may be represented by the following code (The sense of lines of the PROLOG-code is explained on the right side):

```
schedule(X,Y):=
X = [X1,X2,X3,X4,X5],
-          /*list of days of agent
X*/
Y = [Y1,Y2,Y3,Y4,Y5],
-          /*list of days of agent
Y*/
X ins 0..1,
-/*Fuzzy degrees of X-disposition*/
Y ins 0..1,
-/*Fuzzy degrees of Y-disposition */
sum(X, #<, 4),           /*Restriction
-       on X-activity during a week*/
sum(Y, #<, 4),           /*Restriction
-         on Y-activity during a
week*/
X1 # / Y1,# (X1 #/ Y1),
/*Conditions
-      on activity of agents each
day*/
X2 # / Y2,# (X2 #/ Y2),
X3 # / Y3,# (X3 #/ Y3),
X4 # / Y4,# (X4 #/ Y4),
X5 # / Y5,# (X5 #/ Y5),
label([X1,X2,X3,X4,X5,Y1,Y2,Y3,Y4,Y5]).
```

In result, the following list of solutions is obtained:

```
X = [0, 0, 1, 0, 1],
Y = [1, 1, 0, 1, 0] ;
X = [0, 0, 1, 1, 0],
Y = [1, 1, 0, 0, 1] ;
X = [0, 0, 1, 1, 1],
Y = [1, 1, 0, 0, 0] ;
X = [0, 1, 0, 1, 0],
Y = [1, 0, 1, 0, 1];
```

```
X = [0, 1, 0, 1, 1],
Y = [1, 0, 1, 0, 0];
X = [0, 1, 1, 0, 0],
Y = [1, 0, 0, 1, 1];;
X = [0, 1, 1, 0, 1]
Y = [1, 0, 0, 1, 0];
X = [0, 1, 1, 1, 0],
Y = [1, 0, 0, 0, 1] ;
X = [1, 0, 0, 0, 1],
Y = [0, 1, 1, 1, 0] ;
X = [1, 0, 0, 1, 0],
Y = [0, 1, 1, 0, 1] ;
X = [1, 0, 1, 0, 0],
Y = [0, 1, 0, 1, 1] ;
X = [1, 0, 1, 1, 0],
Y = [0, 1, 0, 0, 1] ;
```

It is not difficult to observe that each pair of the achieved solutions respect all the requirements (1 for X admints 0 for Y and conversly).

Case 2 – MAP(2, 6, 4). In this situation, all the earlier requirements remain without changes with exception of the length of a working period for both agents (6 instead of 5) and the admitted number of acitivities during a week (we exchange an upper bound equal to 5 for 4). In this framework, the appropriate PROLOG-schedule is given by the code clauses:

```
schedule(X,Y):=
X = [X1,X2,X3,X4,X5,X6],
Y=[Y1,Y2,Y3,Y4,Y5, Y6],
X ins 0..1,
Y ins 0..1,
sum(X, <, 5),
sum(Y, <, 5),
X1 # / Y1, # (X1 #/  Y1),
X2 # / Y2, # (X2 #/  Y2),
X3 # / Y3, # (X3 #/  Y3),
X4 # / Y4, # (X4 #/  Y4),
X5 # / Y5, # (X5 #/  Y5),
X6 # / Y6, # (X6 #/  Y6),
label([X1,X2,X3,X4,X5, X6, Y1, Y2,
Y3, Y4, Y5, Y6]).
```

In result, we obtain 19 pairs of lists for X and Y. Some of the admitted solutions are listed here:

```
X = [0, 0, 0, 1, 1, 1],
Y = [1, 1, 1, 0, 0, 0] ;
X = [0, 0, 1, 0, 1, 1],
Y = [1, 1, 0, 1, 0, 0] ;
X = [0, 0, 1, 1, 0, 1],
Y = [1, 1, 0, 0, 1, 0] ;
X = [0, 0, 1, 1, 1, 0],
Y = [1, 1, 0, 0, 0, 1] ;
X = [0, 1, 0, 0, 1, 1],
Y = [1, 0, 1, 1, 0, 0].
```

6.4 Fuzzy Cases

In order to exchange the crisp cases for the fuzzy ones it is enough to admit some new degrees of agent dispositions In the current cases, the following list of values is admitted:

0 – to represent the fact that an agent A is absent (on a shift),

1 – to represent a physical absence of the agent A, but a real disposition to be present.

2 – to represent a physical presence of A, which is only in a partial disposition to work.

3 – to represent a full disposition of A to work[13].

Let us begin with an exemplary case of two agents: A1 and A2 working 5 days in a week and having four degrees of disposition denoted by 0,1,2 and 3 as above.

MAP(2,5,4)^{Fuzz}. This situation is defined by the following requirements:

— 2 agents: A1, A2 operating during a 5-day scheduling period,

— 4 degrees of ability to work are associated to the agents: 0, 1, 2, 3.

[13] As mentioned, we rather prefer to think about these values as normalized to [0,1] – as $\frac{1}{3}, \frac{2}{3}$ etc. instead of 1, 2, or 3. We use values 0, 1, 2, 3 because of restrictions imposed on PROLOG-syntax.

— a summaric number of admissible degrees of A1-disposition is smaller than 12,

— a summaric number of admissible degrees of A2-disposition is smaller than 9,

— a summaric restriction on the admissible number of the disposition degrees for both A1 and A2 (together) in the first day is smaller than 4, for A1 (alone) is greather than 1, what excludes a degree of A2-activity greather than 2,

— a summaric restriction on the admissible number of the disposition degrees for both A1 and A2 (together) in the second, the third, the 4th and in the 5th day (D1, D2, D3, D2, D5) is smaller than 4, for A1 (alone) – is greather than 2, what excludes a degree of A2-activity greather than 2,

— a summaric activity of A1 in the day triples: (D1, D2, D3) and (D3,D4,D5) is smaller than 6,

— a summaric activity of A2 in the day triples: (D1, D2, D3) and (D3,D4,D5) is smaller than 7,

— a summaric activity of A1 in a day triple (D2, D2, D4) is smaller than 7,

— a summaric activity of A2 in a day triple (D2, D3, D4) is smaller than 5.

This situation may be reflected in the following PROLOG-program (As earlier, the sense of lines of the PROLOG-code is explained on the right side of the programm):

```
schedule2(A1,A2) :- A1
= [A1D1,A1D2,A1D3,A1D4,A1D5],
-           /* Days of agent A1*/
A2 = [A2D1,A2D2,A2D3,A2D4,A2D5],
-   /* list of days of agant A2 */
A1 ins 0..3,
/* Fuzzy degrees of A1-disposition */
A2 ins 0..3,
/* Fuzzy degrees of A2-disposition */
sum(A1, #<, 12),
-     /* Restriction on a week
activity      of A1 */
sum(A2, #<, 9),
-     /* Restriction on a week
activity of A2 */
sum([A1D1, A2D1], #<, 4), (A1D1 #> 1)
# / (A2D1 #> 2),
-             /* Restriction on D1 */
sum([A1D2, A2D2], #<, 4), (A1D2 #> 2)
# / (A2D2 #> 2),
-             /* Restriction on D2 */
sum([A1D3, A2D3], #<, 4), (A1D3 #> 2)
# / (A2D3 #> 2),
-             /* Restriction on D3 */
sum([A1D4, A2D4], #<, 4), (A1D4 #> 2)
# / (A2D4 #> 2),
-             /* Restriction on D4 */
sum([A1D5, A2D5], #<, 4), (A1D5 #> 2)
# / (A2D5 #> 2),
-             /* Restriction on D5 */
sum([A1D1, A1D2, A1D3], #<, 6),
sum([A1D2, A1D3, A1D4], #<, 7),
sum([A1D3, A1D4, A1D5], #<, 6),
sum([A2D1, A2D2, A2D3], #<, 7),
sum([A2D2, A2D3, A2D4], #<, 5),
sum([A2D3, A2D4, A2D5], #<, 7),
-     /* Restrictions on the next 3
days*/
label([A1D1,A1D2,A1D3,A1D4,A1D5,
A2D1,A2D2,A2D3,A2D4,A2D5]).
```

In this case, the PROLOG-solver returns us the following solution-lists:

```
A1 = [2,3,0,3,0]
A2 = [0,0,3,0,3] and
A1 = [2,3,0,3,0]
A2 = [1,0,3,0,3].
```

MAP(2,5,5)^{Fuzz}. Let us slightly modify temporal conditions imposed on work conditions of agents A1 and A2 as follows:

- instead of 0,1,2,3 we consider five values: 0,1,2,3,4 as admissible fuzzy disposition degrees of A1 and A2,
- restriction on A2-activity during a week is relaxed – instead of `sum(A2, #<, 9)` we adopt a condition `sum(A2, #<, 10)`.

ISSN 2007-9737

Taking into account these requirements, one achieves the following program:

```
schedule2(A1,A2) :-
A1 = [A1D1,A1D2,A1D3,A1D4,A1D5],
-       /* list of days of agent A1*/
A2 = [A2D1,A2D2,A2D3,A2D4,A2D5],
-       /* list of days of agent A2
*/
A1 ins 0..4,
/* Fuzzy degrees of A1-disposition */
A2 ins 0..4,
/* Fuzzy degrees of A2-disposition */
sum(A1, #<, 12),
-      /* Restriction on A1-activity
during a week */
sum(A2, #<, 10),
-      /* Restriction on A2-activity
during a week */
sum([A1D1, A2D1], #<, 4), (A1D1 #> 1)
# / (A2D1 #> 2),
-           /* Restriction on D1 */
sum([A1D2, A2D2], #<, 4), (A1D2 #> 2)
# / (A2D2 #> 2),
-           /* Restriction on D2 */
sum([A1D3, A2D3], #<, 4), (A1D3 #> 2)
# / (A2D3 #> 2),
-           /* Restriction on D3 */
sum([A1D4, A2D4], #<, 4), (A1D4 #> 2)
# / (A2D4 #> 2),
-           /* Restriction on D4 */
sum([A1D5, A2D5], #<, 4), (A1D5 #> 2)
# / (A2D5 #> 2),
-           /* Restriction on D5 */
sum([A1D1, A1D2, A1D3], #<, 6),
sum([A1D2, A1D3, A1D4], #<, 7),
sum([A1D3, A1D4, A1D5], #<, 6),
sum([A2D1, A2D2, A2D3], #<, 7),
sum([A2D2, A2D3, A2D4], #<, 5),
sum([A2D3, A2D4, A2D5], #<, 7),
label([A1D1,A1D2,A1D3,A1D4,A1D5,
-     /* Restrictions on the next 3
days*/
A2D1,A2D2,A2D3,A2D4,A2D5]).
```

In this situation, the PROLOG-solver returns us the following (slightly longer) list of solutions:

```
A1 = [0, 3, 0, 3, 0],
A2 = [3, 0, 3, 0, 3] ;
A1 = [2, 3, 0, 3, 0],
A2 = [0, 0, 3, 0, 3] ;
A1 = [2, 3, 0, 3, 0],
A2 = [1, 0, 3, 0, 3].
```

6.5 A Brief Discussion of the Results: Diagnosis and Prediction

It easy to see that a number of solutions S linearly depends on a number of admitted working days in the considered crisp cases. In fact, 5 working days (with 2 agents) gives us 15 solutions (pairs). It may be surprising that extending a scheduling period to 6 working days gives 19 solutions (by no further changes). It means that $S \sim 3N$, where N denotes a number of working days.

It may be an intriguing fact that exchanging a number of admissible degrees of agent ability (4 for 5) only slightly exchanges the number of solutions – (2 for 3). In fact, we only get 2 solutions (pairs) for $MAP(2,5,4)^{Fuzz}$ and 3 for $MAP(2,5,5)^{Fuzz}$. Obviously, a similar linear relationship holds between each of the input parameters and the number of solutions.

Simoultaneously, if we admit 6 degrees instead of 5 and exchange 4 for 5 in each place of the program for MAP^{Fuzz}(2,5,5) – (without any further modification) the number of solutions radicaly increses – as 16 new solutions are returned. Meanwhile, a slight relaxation of the conditions: `sum(A1, #<, 10)` for `sum(A1, #<, 13)` and `sum(A2, #<, 10)` for `sum(A2, #<, 13)` – preserves the same number of solutions (16).

Further relaxation of temporal constrains usually changes a combinatorial explosion of the algorithm relatively quickly. For example, if exchange also a requirement `sum([A1D1, A1D2, A1D3], #<, 6)` for `sum([A1D1, A1D2, A1D3], #<, 13)`, one gets more than 60 solutions.

7 Conclusion

It has already been shown how FMAP may be put forward in two depictions. The first depiction enables an arithmetic representation of different constraints imposed on FMAP, whereas the second general depiction ellucidates a disctinction criterion between FMAP and the original MAP. Finally, some workable subsaces of FMAP were solved by means of SWI-PROLOG solvers.

Nevertheless, it appears that PROLOG only approximates a kind of fuzziness in the context of MAP. In fact, considering different degrees of agant ability introduces a kind of a multi-valency to different workable subcases of FMAP. In this perspective, Bousi-PROLOG could lay claim to the role of the more adequate language to grasp different FMAP.

Acknowledgment

The authors would like to thank to Krzysztof Kluza. His comments, remarks and discussions on different aspect of the presented issue allowed me to improve the paper.

References

1. **Burke, E. K., Curtois, T., Qu, R., & Berghe, G. V. (2007).** *A scatter search approach to the nurse rostering problem*. Technical report, School of Computer Science and Information Technology, University of Nottingham.
2. **Cetin, E. & Sarucan, A. (2014).** Nurse scheduling using binary fuzzy goal programming. *Proceedings of ICAMSAO*, pp. 1–6.
3. **Cheang, B., Li, H., Lim, A., & Rodrigues, B. (2003).** Nurse rostering problems–a bibliographic survey. *European Journal of Operational Research*.
4. **Dechter, R., Meiri, & Pearl, J. (1991).** On fuzzy temporal constraints networks. *Temporal constraints network*, Vol. 49(1-3), pp. 61–95.
5. **Ernst, A. T., Jiang, H., Krishnamoorthy, M., & Sier, D. (February 2004).** Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, Vol. 153(1), pp. 3–27.
6. **Felici, G. & Gentile, C. (2004).** A polyhedral approach for the staff rostering problem. *Management Science*, Vol. 50(3), pp. 381–394.
7. **Iranzo, P. & Rubio-Manzano, C. (2010).** Bousi-prolog: A fuzzy programming language for modelling vague knowledge and approximate reasoning. *IJCCI*, pp. 93–98.
8. **Iranzo, P. & Rubio-Manzano, C. (2010).** A programming environment for bousi-prolog. *IC-AI*, pp. 36–42.
9. **Jobczyk, K. (2017).** *Temporal Planning with Fuzzy Constraints and Preferences*. PhD thesis, Normandy University.
10. **Jobczyk, K. & Ligeza, A. (2016).** Multi-valued halpern-shoham logic for temporal allen's relations and preferences. *Proceedings of the annual international conference of Fuzzy Systems (FuzzIEEE)*.
11. **Jobczyk, K. & Ligeza, A. (2016).** Towards a new convolution-based approach to the specification of stpu-solutions. *FUZZ-IEEE*, pp. 782–789.
12. **Jobczyk, K. & Ligeza, A. (2017).** Dynamic epistemic preferential logic of action. *ICAISC*, Vol. 2, pp. 243–254.
13. **Jobczyk, K., M., B., Ligeza, A., & Karczmarczuk, J. (2014).** Fuzzy integral logic expressible by convolutions. *Proceeding of ECAI'2014*, pp. 1042–1043.
14. **Jobczyk, K., M., B., Ligeza, A., & Karczmarczuk, J. (2015).** Comparative approach to the multi-valued logic construction for preferences. *Proceeding of ICAISC'15*, Vol. LNAI.
15. **Khatib, L., Morris, P., Morris, R., & Rossi, F. (2001).** Temporal reasoning about preferences. *Proceedings of IJCAI-01*, pp. 322–327.
16. **Leierson, C. & Saxe, J. (1983).** A mixed-integer linear programming problem which is efficiently solvable. *Proceedings 21st Annual Alerton Conference on Communications, Control, and Computing*, pp. 204–213.
17. **Lepegue, T., Prot, D., & Bellenguez-Morineau, O. (2012).** A tour scheduling problem with fixed jobs: use of constraint programming. *Practice and Theory of Automated Timetabling*, Vol. August, pp. 29–31.
18. **Liao, Y. & Wong, C. (1983).** An algorithm to compact a vlsi symbolic layout with mixed constraints. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 2(2), pp. 62–69.

19. **Liu, J., Jing, H., & Tang, Y. (2002).** Multi-agent oriented constraints satisfaction. *Artificial Intelligence*, Vol. 136, No. 1, pp. 101–144.
20. **Mailler, R. & Lesser, V. (2006).** Asynchronous partial overlay: A new algorithm for solving distributed constraints satisfaction problems. *Journal of Artificial Intelligence Research*, Vol. 25, pp. 529–576.
21. **Metivier, J.-P., Boizumault, P., & Loudni, S. (2009).** Solving nurse rostering problems using soft global constraints. *Proceedings of the 15th international conference on Principles and practice of constraint programming, CP'09*, pp. 73–87.
22. **Modi, P., Shen, W., Tambe, M., & Yakoo, M. (2005).** Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, Vol. 161, pp. 149–180.
23. **Qu, R. & He, F. (2008, Nottingham).** A hybrid constraint programming approach for nurse rostering prolems. *Technical report, School of Computer Science*.
24. **Rossi, F., Yorke-Smith, N., & Venable, K. (2003).** Temporal reasoning with preferences and uncertainty. *Proceedings of AAAI*, volume 8, pp. 1385–1386.
25. **Selim, H. & Topaloglu (2004).** Nurse scheduling using modeling approach. *Fuzzy Sets and Systems*, Vol. 161(11), pp. 1543–1563.
26. **Shostak, R. (1981).** Deciding linear inequalities by computing loop residues. *Journal of ACM*, Vol. 28(4), pp. 769–779.
27. **Spivak, D. (2014).** *Category theory for the sciences*. Cambridge Press.
28. **Traverso, P., Ghallab, M., & Nau, D. (1997).** *Automated Planning: theory and practice*, volume 2004. Elsevier.
29. **Vidal, T. & Fargier, H. (1999).** Handling contingency in temporal constraints networks: From consistency to controllabilities. *Journal of Experimental and Technical Artificial Intelligence*, Vol. 11, No. 1, pp. 23–45.
30. **Yokoo, M. (2001).** *Distributed Constraints Satisfaction: Foundation of Cooperation in Multi-Agent System*. Springer.

Article received on 29/10/2019; accepted on 07/03/2020.
Corresponding author is Krystian Jobczyk.