Modeling Fuzzy Intervals With Constraint Logic Programming

Przemysław Kobylański

Wrocław University of Technology, Institute of Mathematics and Computer Science, Wrocław, Poland, email: Przemyslaw.Kobylanski@pwr.wroc.pl

Abstract. In this paper the method of modeling fuzzy intervals in fuzzy decision-making is presented. Described method makes use of constraint logic programming and it is based on the concept of descriptors. This approach is very general and it is consistent with Zadeh's extension principle and Bellman-Zadeh concept of fuzzy decision making. It fulfills Klir's requisite constraint and deals effectively with a drowning effect too. The idea of descriptors of fuzzy intervals and fuzzy constraints is illustrated with computational example of flexible scheduling problem in which robust for drowning effect schedule is found.

1 Introduction

It is important to develop tools for fuzzy decision-making which will be consistent with the fundamental principles like the Zadeh's extension principle, the Bellman-Zadeh concept of fuzzy decision-making [1], the Klir's requirement for requisite constraints [8] and Dubois and Fortemps remarks on the drowning effect related with sup-min criterion [5].

The paper presents one of such tool which fulfills all of above assumptions. CLP(F) is a module for modeling and solving fuzzy decision-making in declarative programming language Prolog. Homepage of our project is http://www.im.pwr.wroc.pl/~przemko/clpf and there are sources of CLP(F) module for some implementations of Prolog such as Ciao Prolog, IF/Prolog, SICStus Prolog and YAP Prolog. The module CLP(F) is based on the concept of descriptors [11] which are implemented with constraint logic programming. There are a number of advantages of using CLP(F) module e.g. symbolic computations and full information about all solutions (feasible, optimal and improved optimal).

The paper is organized as follows: Section 2 describes basics of declarative programming paradigm and the elements of constraint programming, Section 3 describes the fuzzy constraint satisfaction problem, next section shows how to model and solve fuzzy decision problems using constraint programming and fuzzy intervals, fuzzy constraints and systems of constraints represented in the form of descriptors, Section 5 presents computational example in which the improved schedule for flexible scheduling problem from [5] is computed with the proposed CLP(F) module.

2 Declarative and constraint programming

In an imperative (algorithmic) paradigm the problem is solved by giving the algorithm (procedure) which solves it. A declarative paradigm allows us to solve the problem in the other, much simpler, way. It is possible to state the problem and computer finds the solution based on this description.

2.1 Programming in logic

One of the most popular declarative programming language is Prolog (PROgramming in LOGic). In this language, the problem is described with logical clauses from the first-order predicate calculus. Program consists of facts and rules which states the relations over the domain of considered problem.

Example 2.1. Let us consider the following example of concatenation of two lists. In the algorithmic programming we think about concatenation as the process of searching for the end of the first list and binding the first element of the second list just after the last element of the first list. It is impossible to make use of the same procedure for splitting the giving list into two parts. In the declarative programming we consider concatenation as the relation between three lists: L1, L2 and a list which is concatenations of L1 and L2. Prolog program for concatenation of lists consists of the following two clauses:

```
append([], L, L).
append([H | T], L, [H | TL]) :- append(T, L, TL).
```

The first fact states that concatenation of empty list [] with any list L is equal to L. The second clause states that if TL is concatenation of T and L, then the list [H | TL] with head H and tail TL is concatenation of lists [H | T] and L (symbol :- is read as *if*).

Prolog makes use of SLD-resolution (*Selection rule driven Linear resolution for Defined clauses*) for derivation of answers for the given goal [14]. The number of answers depends on the number of possible proofs (SLD-derivations). All SLD-derivations are collected as branches in the SLD-tree (see Figure 1).

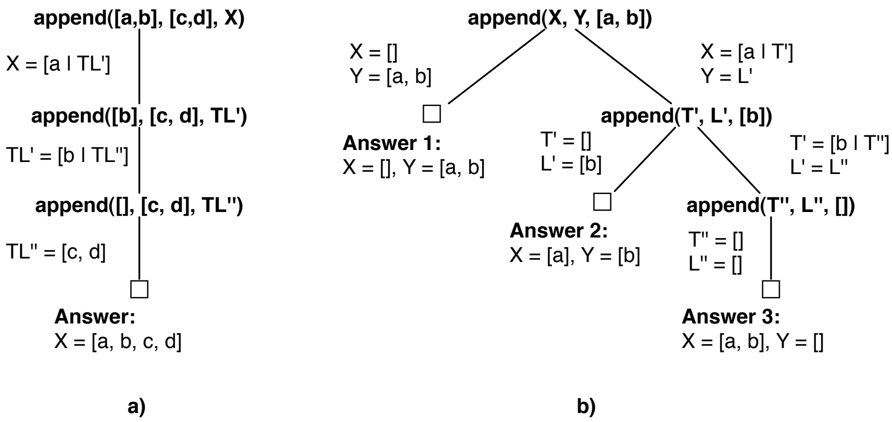

Figure 1. SLD-trees for: a) concatenating [a,b] and [c,d], b) splitting [a,b].

The goal is placed in the root of the SLD-tree. Each edge in the tree is labeled with proper substitution. All branches ended with symbol □, which represents empty goal, lead to the answers which are the compositions of all substitutions on the branch. In the

SLD-tree presented in Figure 1a) there is one answer [a,b,c,d] because there is only one concatenation of lists [a,b] and [c,d]. In Figure 1b) there are three answers because list of two elements [a, b] could be split in three different ways: [], [a,b]; [a], [b]; [a,b], [].

In general case, many branches in the SLD-tree end with failure and do not lead to any answer (solution). An SLD-tree sometimes contains infinite branches and the SLD-resolution, which is depth-first searching of SLD-tree, does not finish computations.

2.2 Constraint logic programming

As it was shown, the process of solving the stated problem in programming in logic is depth-first searching of some SLD-tree. The efficiency of searching depends on how early the procedure finds out that considered SLD-subtree does not contain any branch which leads to solution.

The most powerful method for improving the efficiency of searching for solutions in SLD-tree is constraint technology. Opposite to standard of Prolog, it allows imposing constraints over uninstantiated variables (variables without assigned values). During propagation of such constraints system could detect inconsistency and stop searching in considered branch. In this case backtrack is done and searching is continued in the next branch.

Methods for processing of constraints are described in [2].

Discrete case. In this case some variables have assigned domains i.e. finite subsets of integers which contains all possible values for this variables. During propagation of constraints domains are restricted by excluding the values which do not satisfy some of constraints. In many problems the propagation of constraints does not ensure detection of inconsistency. After constraints propagation labeling of variables is done by assigning (with backtracks) the values from domains to variables.

Example 2.2. The following goal checks if it is possible to choose three pairwise different values from the set $\{1, 2\}$, where `#\=` is *not equal* relation:

```
?- [X, Y, Z] in 1..2, X #\= Y, X #\= Z, Y #\= Z.
yes
```

The positive answer is incorrect and the labeling is necessary to detect inconsistency:

```
?- [X, Y, Z] in 1..2, X #\= Y, X #\= Z, Y #\= Z, labeling([X, Y, Z]).
no
```

The family of programming languages which enable such constraints is called CLP(FD) (*constraint logic programming over finite domains*).

Continuous case. In this case it is possible to impose constraints over uninstantiated variables with values from the set of rational or real numbers [7]. The Fourier-Motzkin algorithm is used for projection of linear inequalities and simplex method is used for optimizing linear objective functions.

Example 2.3. Let us consider the following very simple linear programming model:

```
?- {X >= 0, Y >= 0, 2*X + 3*Y =< 10, 3*X =< 5, Z = X+Y}, maximize(Z).
X = 5/3,
Y = 20/9,
Z = 80/9
```

The family of programming languages which enable such constraints is called CLP(Q) or CLP(R) (*constraint logic programming over rational numbers* or *over real numbers*).

3 Fuzzy constraint satisfaction problem

Uncertainty could be expressed in many ways and one of them is the theory of fuzzy sets [6].

Definition 3.1. A fuzzy set F is equivalent to giving the reference set Ω and a mapping, μ_F, of Ω into $[0,1]$. The value $\mu_F(\omega)$, for $\omega \in \Omega$, is interpreted as the degree of membership of ω in the fuzzy set F. For the giving value $\lambda \in (0,1]$ the set

$$F^\lambda = \{\omega \in \Omega | \mu_F(\omega) \geq \lambda\} \tag{1}$$

is called a λ-cut of F.

For a given fuzzy set F, the family of all λ-cuts is monotone i.e.

$$0 < \lambda_1 \leq \lambda_2 \leq 1 \rightarrow F^{\lambda_2} \subseteq F^{\lambda_1}. \tag{2}$$

Definition 3.2. Let $A_1, \ldots, A_n$ be fuzzy sets in $\Omega_1, \ldots, \Omega_n$ with the membership functions $\mu_{A_1}(x), \ldots, \mu_{A_n}(x)$, respectively. Then the Cartesian product of the fuzzy sets $A = A_1 \times \cdots \times A_n$ is defined as a fuzzy set in $\Omega = \Omega_1 \times \cdots \times \Omega_n$ whose membership function has the following form:

$$\begin{aligned} \mu_A(\boldsymbol{x}) &= \mu_{A_1 \times \cdots \times A_n}(x_1, \ldots, x_n) \\ &= \min\{\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n)\}. \end{aligned} \tag{3}$$

Let us recall the extension principle of Zadeh [15], which provides a general method for extending a non-fuzzy mathematical concept to the fuzzy framework:

Definition 3.3. Let f be be mapping from Ω to a set $\mathbb{Y}$ such that $y = f(\boldsymbol{x})$. Then the fuzzy set B in $\mathbb{Y}$ induced by the fuzzy set A in Ω is defined by the following membership function:

$$\mu_B(y) = \begin{cases} \sup_{\substack{\boldsymbol{x} \in \Omega \\ y = f(\boldsymbol{x})}} \mu_A(\boldsymbol{x}) & \text{for} \quad f^{-1}(y) \neq \emptyset, \\ 0 & \text{for} \quad f^{-1}(y) = \emptyset. \end{cases} \tag{4}$$

Nguyen [13] proposed an equivalent representation of the extension principle:

Theorem 3.4. *If there exist $x_1, \ldots, x_n$ such that $\mu_B(y) = \mu_{A_1 \times \cdots \times A_n}(x_1, \ldots, x_n)$ for any $y \in \mathbb{Y}$ (the supremum of (4) is attained in $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$), then the following equality holds:*

$$B^\lambda = [f(A_1, \ldots, A_n)]^\lambda = f(A_1^\lambda, \ldots, A_n^\lambda), \tag{5}$$

where A_i^λ is λ-cut of the fuzzy set A_i.

When the reference set Ω is equal to the set of real numbers a fuzzy set is called a fuzzy quantity.

The special kind of fuzzy quantity is a fuzzy interval i.e. a fuzzy quantity whose membership function is quasiconcave:

$$\forall u \leq v \; \forall w \in [u, v] \; \mu_Q(w) \geq \min(\mu_Q(u), \mu_Q(v)). \tag{6}$$

In many practical applications a fuzzy decision-making problem is formulated as the following fuzzy constraint satisfaction problem [3, 4, 5]:

Definition 3.5. The Fuzzy Constraint Satisfaction Problem (FCSP for short) $\mathscr{P} = (\mathscr{X}, \mathscr{D}, \mathscr{C}, \mathscr{R})$ is

- A set of variables $\mathscr{X} = \{X_1, \ldots, X_n\}$.
- A set of definition domains $\mathscr{D} = \{D_1, \ldots, D_n\}$ where D_i is the definition domain of X_i. Ω is the Cartesian product of the definition domains $\Omega = D_1 \times \cdots \times D_n$.
- A set of constraints $\mathscr{C} = \{C_1, \ldots, C_m\}$. They can be either flexible or classical.
- A set of fuzzy relation $\mathscr{R} = \{R_1, \ldots, R_m\}$ where R_j defines the solutions satisfying more or less the constraint C_j. For the classical constraints, the relation R_j is all-or-nothing, while for the flexible constraints R_j is a fuzzy relation.

For each solution $\overline{d} \in \Omega$, the global satisfaction degree is

$$\mathrm{Sat}(\overline{d}) = \mu_{R_1 \cap \cdots \cap R_m}(\overline{d}) = \min_{C_i \in \mathscr{C}} \mu_{R_i}(\overline{d}). \tag{7}$$

$\mathrm{Sat}(\overline{d})$ gives the degree to which $\overline{d}$ belongs to the fuzzy set of the feasible solutions of $\mathscr{P}$, $\mathrm{Sols}(\mathscr{P})$:

$$\mu_{\mathrm{Sols}(\mathscr{P})}(\overline{d}) = \mathrm{Sat}(\overline{d}). \tag{8}$$

The consistency degree of FCSP is defined as the satisfaction of its best solutions:

$$\mathrm{Cons}(\mathscr{P}) = \sup_{\overline{d} \in \Omega} \mu_{\mathrm{Sols}(\mathscr{P})}(\overline{d}) = \sup_{\overline{d} \in \Omega} \mathrm{Sat}(\overline{d}) = \tag{9}$$

$$= \sup_{\overline{d} \in \Omega} \min_{C_i \in \mathscr{C}} \mu_{R_i}(\overline{d}).$$

The above formulation is consistent with the well known Bellman-Zadeh concept of fuzzy decision-making [1] which combines the multi-objective fuzzy optimization and fuzzy constraint satisfaction problems into one fuzzy constraint satisfaction problem.

The consistency degree of FCSP is obtained for the solution $\overline{d} \in \Omega$ which maximizes the value $\min_{C_i \in \mathscr{C}} \mu_{R_i}(\overline{d})$. The sup-min criterion does not ensure the maximal value for each $\mu_{R_i}(\overline{d})$ and probably some of them could be improved. This phenomenon is called the drowning effect. If the linear programming is used to determine the best solution according to global satisfaction degree, the solution $\overline{d}$ is at facet of the feasible region (at the ends of the proper α-cuts) and all the constraints $C_i \in \mathscr{C}$ are satisfied at the same degree equal to $\mathrm{Cons}(\mathscr{P})$.

The drowning effect and the methods for improving solutions are described in [5].

4 Modeling fuzzy constraints

In [11] modeling fuzzy constraints with constraint programming was proposed. The fuzzy intervals, expressions, constraints and systems of constraints are represented with special structures called descriptors. The descriptors contain real or rational variables constrained with system of linear equalities and inequalities.

4.1 Descriptors

A fuzzy interval is represented by the region under the plot of its membership function. This region is described with a set of constraints:

Definition 4.1. Let interval $[\underline{F}^{\lambda}, \overline{F}^{\lambda}]$ be the λ-cut of fuzzy interval $\widetilde{F}$. The descriptor $D(\widetilde{F})$ of fuzzy interval $\widetilde{F}$ is a structure:

$$(x, \lambda)@C(x, \lambda), \tag{10}$$

where $C(x, \lambda)$ is the system of constraints restricting the values of variables $x, \lambda \in \mathbb{R}$:

$$C(x, \lambda) = \{0 \leq \lambda \leq 1, \underline{F}^{\lambda} \leq x \leq \overline{F}^{\lambda}\}. \tag{11}$$

Descriptor of the value of a fuzzy arithmetic expression could be build with the descriptors of its subexpressions:

Definition 4.2. Let $D(\widetilde{V}_i) = (x_i, \lambda_i)@C_i(x_i, \lambda_i)$, for $i \in \{1, 2\}$, be descriptors of fuzzy values $\widetilde{V}_i$ of two expressions. Then descriptor $D(\widetilde{V}_1 \tilde{\diamond} \widetilde{V}_2)$ of the fuzzy value of expression $\widetilde{V}_1 \tilde{\diamond} \widetilde{V}_2$ has the following form:

$$\begin{gathered}(x, \lambda)@C_1(x_1, \lambda_1) \cup C_2(x_2, \lambda_2) \cup \\ \{\lambda \leq \lambda_1, \lambda \leq \lambda_2, x = x_1 \diamond x_2\}.\end{gathered} \tag{12}$$

A fuzzy constraint is represented with a descriptor build with two descriptors of the value of fuzzy expressions:

Definition 4.3. Let $D(E_i) = (x_i, \lambda_i)@C_i(x_i, \lambda_i)$, where $i \in \{1, 2\}$, are two descriptors of two values of fuzzy expressions E_1 and E_2. Then, for a relation $\propto \in \{=, \leq, \geq, <, >, \neq\}$, descriptor $D(E_1 \tilde{\propto} E_2)$ of fuzzy constraint $E_1 \tilde{\propto} E_2$ has the following form:

$$\begin{gathered}(\langle x_1, x_2 \rangle, \lambda)@C_1(x_1, \lambda_1) \cup C_2(x_2, \lambda_2) \cup \\ \{\lambda \leq \lambda_1, \lambda \leq \lambda_2, x_1 \propto x_2\}.\end{gathered} \tag{13}$$

Descriptor of the system of fuzzy constraints is build with the descriptors of fuzzy constraints:

Definition 4.4. Let $S = \{C_1, C_2, \ldots, C_n\}$ be the system of fuzzy constraints $C_1, C_2, \ldots, C_n$ and $D(C_i) = (\boldsymbol{x}^{(i)}, \lambda_i)@C_i(\boldsymbol{x}^{(i)}, \lambda_i)$, where $i = 1, 2, \ldots, n$. Then descriptor $D(S)$ has the following form:

$$\begin{gathered}(\boldsymbol{x}^{(1)}; \boldsymbol{x}^{(2)}; \ldots; \boldsymbol{x}^{(n)}, \lambda)@\bigcup_{i=1}^{n} C_i(\boldsymbol{x}^{(i)}, \lambda_i) \cup \\ \{\lambda \leq \lambda_1, \lambda \leq \lambda_2, \ldots, \lambda \leq \lambda_n\},\end{gathered} \tag{14}$$

where

$$\langle x_1, \ldots, x_n \rangle; \langle y_1, \ldots, y_m \rangle = \langle x_1, \ldots, x_n, y_1, \ldots, y_m \rangle.$$

In [11] propositions that descriptors are consistent with Zadeh's extension principle are proved.

4.2 Constraint networks and improving optimal solution

When the descriptor is used a suitable constraint network is created (about constraint networks and constraint processing see [2]). It contains nodes for each decision variable and the arcs corresponding to imposed constraints.

In Figure 2 an example of descriptor $D(\{\widetilde{X}\tilde{\leq}\widetilde{Z}, \widetilde{Y}\tilde{\leq}\widetilde{Z}\})$ is presented. It is worth noticing that the box for descriptor of fuzzy interval $\widetilde{Z}$ is simultaneously nested in the box for descriptor of constraint $\widetilde{X}\tilde{\leq}\widetilde{Z}$ and in the box for descriptor of constraint $\widetilde{Y}\tilde{\leq}\widetilde{Z}$. This ensures that realization z of fuzzy interval $\widetilde{Z}$ is the same in both constraints.

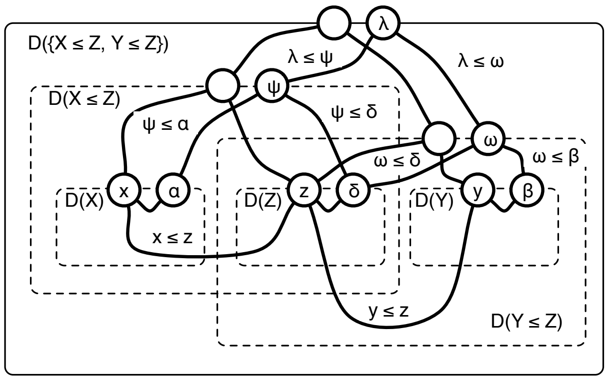

Figure 2. Constraint network for $D(\{\widetilde{X}\tilde{\leq}\widetilde{Z}, \widetilde{Y}\tilde{\leq}\widetilde{Z}\})$.

In [9], another method for improving optimal solution with constraint programming was proposed. In the following example the drowning effect is presented an the idea of improving solution is shown.

Example 4.5 (Drowning effect)**.** Let $\widetilde{X} = (0, 1, 2), \widetilde{Y} = (2, 3, 4), \widetilde{Z} = (1, 2, 3)$ are the triangular fuzzy intervals. The set of constraints consists of $\widetilde{X} \leq \widetilde{Z}$ and $\widetilde{Y} \leq \widetilde{Z}$. The constraint network for considered set of constraints is presented in Figure 2. The optimal solution is found with the linear programming (maximization of λ subject to constraints presented in Figure 2). One of the basic optimal solution is $\langle x, y, z\rangle = \langle 0.5, 2.5, 2.5\rangle$ and $\mu_{\widetilde{X}}(x) = \mu_{\widetilde{Y}}(y) = \mu_{\widetilde{Z}}(z) = 0.5$ (see Figure 3a). But this optimal solution could be improved. The better optimal solution is $\langle x', y, z\rangle = \langle 1, 2.5, 2.5\rangle$ with $\mu_{\widetilde{X}}(x') = 1 > 0.5$ (see Figure 3b).

Improving solution could be written in imperative way as Algorithm 1. The procedure *maximize* finds the maximal value for its argument and the predicate *var* is satisfied if its argument is still not fixed decision variable.

The solution found by Algorithm 1 is Pareto-optimal and has the following property:

Property 1. Let $\mathbf{u} = (u_1, \ldots, u_n)$ be the vector of values of variables $\lambda_1, \ldots, \lambda_n$ found by Algorithm 1. If $\mathbf{v} = (v_1, \ldots, v_n)$ is any other vector of feasible values of variables $\lambda_1, \ldots, \lambda_n$, then if for some variable λ_i condition $v_i > u_i$ is fulfilled, then for some other variable λ_j condition $v_j < u_j \leq u_i$ is fulfilled.

The proof of the above property is presented in [9].

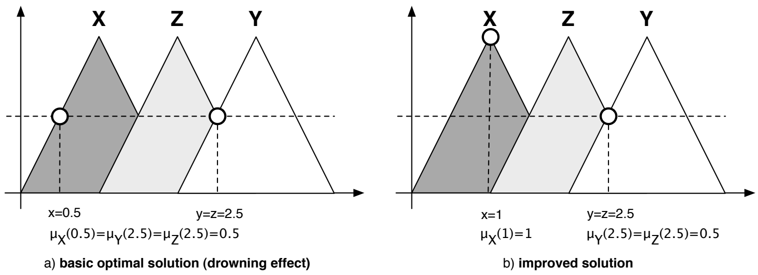

Figure 3. Drowning effect.

Data: Variables for satisfaction degrees $\lambda_1, \lambda_2, \ldots, \lambda_n$
Result: Improved optimal solution
$L_0 \leftarrow \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$;
$k \leftarrow 0$;
while $L_k \neq \emptyset$ **do**
  let y_k be a new variable;
  foreach $\lambda \in L_k$ **do**
    impose constraint $y_k \leq \lambda$;
  end
  call *maximize*(y_k);
  $L_{k+1} \leftarrow \{\lambda \in L_k | var(\lambda)\}$;
  $k \leftarrow k + 1$;
end

Algorithm 1: Improving optimal solution.

Property 1 means that in the solution found by Algorithm 1 it is not possible to increase some satisfaction degree without decreasing some other one. Such solutions are called improved solutions.

Example 4.6 (Improving solution with Algorithm 1)**.** Let $\widetilde{X}, \widetilde{Y}, \widetilde{Z}$ are the triangular fuzzy intervals from Example 4.5. The constraint network corresponding with the descriptor $D(\{\widetilde{X} \tilde{\leq} \widetilde{Z}, \widetilde{Y} \tilde{\leq} \widetilde{Z}\})$ is presented in Figure 2. The degrees for each fuzzy intervals α, β, δ, fuzzy constraints ψ, ω and the system of constraints λ are collected in one set L_0. In the first iteration the following model is solved:

$$y_0 \mapsto \max,$$

$$y_0 \leq \alpha,\ y_0 \leq \beta,\ y_0 \leq \delta,\ y_0 \leq \psi,\ y_0 \leq \omega,\ y_0 \leq \lambda,$$

and variables $\beta, \delta, \psi, \omega, \lambda$ are fixed to $\frac{1}{2}$, $L_1 = \{\alpha\}$. In the second iteration the following model is solved:

$$y_1 \mapsto \max,$$

$$y_1 \leq \alpha,$$

and variable α is fixed to 1, $L_2 = \emptyset$. The condition $L_k \neq \emptyset$, for $k = 2$, in line 3 of the algorithm is false and the algorithm is stopped.

4.3 CLP(F) module

The descriptors were implemented as the CLP(F) module. It makes use of the clp(Q, R) modules [7] for the constraint programming on the rational and real numbers.

CLP(F) module contains predicates for defining fuzzy intervals (*fuzzy*), fuzzy constraints and systems of fuzzy constraints (*fcon*) and predicates for solving the fuzzy constraint satisfaction problem and for computing the consistency degree and improving optimal solution (*cons*). The computations could be done on rational numbers (slower but exact result) or on the real numbers (faster but not exact result).

The fuzzy intervals are defined with predicate *fuzzy(Var, Shape)*, which has two arguments: the variable which is unified with new descriptor and the term which describes the shape of the fuzzy interval. In Figure 4 the available shapes of fuzzy intervals are presented. The parameters A, B, C, D are either rational or real numbers, where $A \leq B \leq C \leq D$.

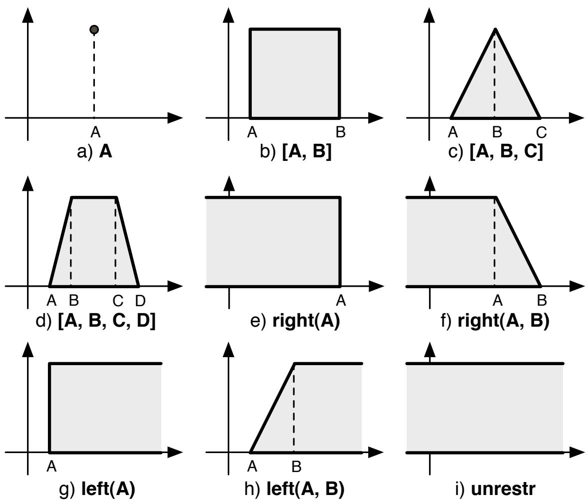

Figure 4. Terms and corresponding shapes of fuzzy intervals.

The fuzzy interval could also be defined as the value of the fuzzy arithmetical expression with predicate *fuzzy(Var, Expr1 OP Expr2)*, where *Var* is the variable unified with new descriptor, *Expr1* and *Expr2* are two fuzzy arithmetical subexpressions and *OP* is one of the following operation: $+$, $-$, $*$, $/$.

In the following example the subtraction of two fuzzy intervals X and Y with the same membership functions is considered:

```
?- fuzzy(X, [1, 2, 3]), fuzzy(Y, [1, 2, 3]),
   fuzzy(Z, X-Y).
X = f(A, B),
Y = f(C, D),
Y = f(E, F),
{E = A-C, D =< 1, D+C =< 3, B+A =< 3,
 D-C =< -1, B-A =< -1, D >= 0, B >= 0,
```

```
 D-F >= 0, B-F >= 0}
```

In the answer the term $f(A, B)$ describes fuzzy interval with realization equal to A and the possible degree of satisfaction of this realization equal B. Between parentheses { and } the system of crisp constraints on realizations and degrees of satisfactions of X, Y and Z is shown.

In the following example the subtraction of two the same fuzzy intervals is considered:

```
?- fuzzy(X, [1, 2, 3]), fuzzy(Y, X),
   fuzzy(Z, X-Y).
X = f(A, B),
Y = f(A, B),
Z = f(0, C),
{B+A =< 3, B-A =< -1, B >= 0, B-C >= 0}
```

Variable Z is equal to 0 according to the requisite constraints [8].

Fuzzy constraint on fuzzy intervals *X1, X2, ... Xn* is defined with the predicate *fcon(Var, [X1, X2, ..., Xn], Constr)*, where *Var* is variable unified with new descriptor and *Constr* is fuzzy constraint or the system of fuzzy constraints build with the relations: $=, \neq, <, \leq, >, \geq$.

The consistency degree *CD* of fuzzy constraint *FC* is computed with the predicate *cons(FC, CD)* and the improved solution, represented by vector *VEC*, is found with the predicate *cons(FC, CD, VEC)*.

The following dialog solves the problem from Example 4.6:

```
?- fuzzy(X, [0, 1, 2]), fuzzy(Y, [2, 3, 4]),
   fuzzy(Z, [1, 2, 3]),
   fcon(FC, [X, Y, Z], (X =< Z, Y =< Z)),
   cons(FC, CON, VEC).
CON = 1/2,
VEC = [f(1,1),f(5/2,1/2),f(5/2,1/2)]
```

Variable *CON* is equal to the consistency degree of fuzzy constraint *FC* and the list *VEC* contains the improved optimal solution (X is equal to 1 with membership function equals to 1, Y and Z are equal to 5/2 with membership functions equal to 1/2).

The constraint networks are created only for the computing consistency degree and after that they are destroyed (no side effects):

```
?- fuzzy(X, [1, 3, 6]), fcon(FC1, [X], X = 2),
   fcon(FC2, [X], X = 4),
   cons(FC1, CON1), cons(FC2, CON2).
CON1 = 1/2,
CON2 = 2/3
```

The constraint $X = 2$ is active only during computation consistency degree *CON1* and it is possible to compute consistency degree *CON2* of constraints $X = 4$.

Table 1. Parameters of operations.

Operation	Ready date	Due date	Duration	Machine
A_1	$(0, 5, \infty, \infty)$	$(-\infty, -\infty, 20, 24)$	$(4, 5, \infty, \infty)$	M_1
A_2			$(3, 4, \infty, \infty)$	M_2
B_1	$(0, 5, \infty, \infty)$	$(-\infty, -\infty, 20, 24)$	$(2, 3, \infty, \infty)$	M_2
B_2			$(1, 2, \infty, \infty)$	M_1
C_1	$(0, 5, \infty, \infty)$	$(-\infty, -\infty, 24, 30)$	$(2, 3, \infty, \infty)$	M_1
C_2			$(8, 9, \infty, \infty)$	M_3
C_3			$(4, 5, \infty, \infty)$	M_2
D_1	$(0, 5, \infty, \infty)$	$(-\infty, -\infty, 24, 30)$	$(7, 8, \infty, \infty)$	M_3
D_2			$(8, 9, \infty, \infty)$	M_1
E_1	$(0, 5, \infty, \infty)$	$(-\infty, -\infty, 24, 30)$	$(0, 1, \infty, \infty)$	M_2
E_2			$(6, 7, \infty, \infty)$	M_3
E_3			$(7, 8, \infty, \infty)$	M_1
E_4			$(2, 3, \infty, \infty)$	M_3

5 Computational example

Let us consider the following example from [5]. In Table 1 the fuzzy ready dates, due dates and durations of operations on machines M_1, M_2, M_3 are presented (all parameters are fuzzy trapezoid intervals). All operations in task have the same ready date and the same due date.

In Figure 5 the precedence relation between operations is presented (this relation is given).

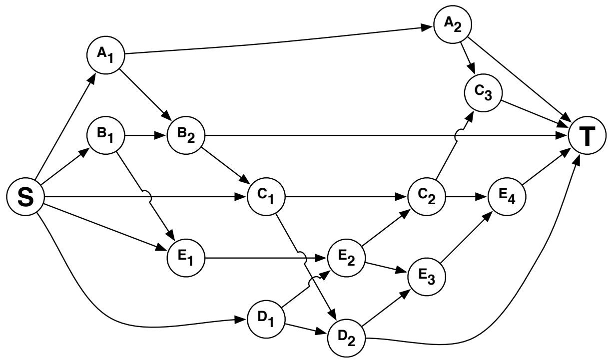

Figure 5. Precedence relation.

The ready dates and duration times are modeled by fuzzy intervals described with term *left(A, B)* and the due dates are described with terms *right(A, B)*. For all task with fuzzy ready date *RD*, fuzzy due date *DD* and fuzzy duration time *DU*, the fuzzy constraints *Start* >= *RD* and *Start* + *DU* =< *DD* are imposed, where *Start* is the start time of operation and it is modeled with fuzzy interval described with term *unrestr* (see Figure 4).

Table 2. Improved schedule

Operation	RD	$Start$	DU	DD
A_1	$\langle\frac{30}{17},\frac{6}{17}\rangle$	$\langle\frac{30}{17},1\rangle$	$\langle\frac{74}{17},\frac{6}{17}\rangle$	$\langle x_1,1\rangle$
A_2	$\langle x_2,1\rangle$	$\langle x_3,1\rangle$	$\langle x_4,1\rangle$	$\langle x_5,1\rangle$
B_1	$\langle\frac{175}{51},\frac{35}{51}\rangle$	$\langle\frac{175}{51},1\rangle$	$\langle\frac{137}{51},\frac{35}{51}\rangle$	$\langle x_6,1\rangle$
B_2	$\langle x_7,1\rangle$	$\langle\frac{104}{17},1\rangle$	$\langle\frac{23}{17},\frac{6}{17}\rangle$	$\langle x_8,1\rangle$
C_1	$\langle x_9,1\rangle$	$\langle\frac{127}{17},1\rangle$	$\langle\frac{40}{17},\frac{6}{17}\rangle$	$\langle x_{10},1\rangle$
C_2	$\langle x_{11},1\rangle$	$\langle\frac{46}{3},1\rangle$	$\langle\frac{25}{3},\frac{1}{3}\rangle$	$\langle x_{12},1\rangle$
C_3	$\langle x_{13},1\rangle$	$\langle\frac{71}{3},1\rangle$	$\langle\frac{13}{3},\frac{1}{3}\rangle$	$\langle 28,\frac{1}{3}\rangle$
D_1	$\langle\frac{5}{3},\frac{1}{3}\rangle$	$\langle\frac{5}{3},1\rangle$	$\langle\frac{22}{3},\frac{1}{3}\rangle$	$\langle x_{14},1\rangle$
D_2	$\langle x_{15},1\rangle$	$\langle\frac{167}{17},1\rangle$	$\langle\frac{142}{17},\frac{6}{17}\rangle$	$\langle x_{16},1\rangle$
E_1	$\langle x_{17},1\rangle$	$\langle x_{18},1\rangle$	$\langle x_{19},1\rangle$	$\langle x_{20},1\rangle$
E_2	$\langle x_{21},1\rangle$	$\langle 9,1\rangle$	$\langle\frac{19}{3},\frac{1}{3}\rangle$	$\langle x_{22},1\rangle$
E_3	$\langle x_{23},1\rangle$	$\langle\frac{309}{17},1\rangle$	$\langle\frac{125}{17},\frac{6}{17}\rangle$	$\langle\frac{434}{17},\frac{38}{51}\rangle$
E_4	$\langle x_{24},1\rangle$	$\frac{434}{17},1\rangle$	$\langle\frac{40}{17},\frac{6}{17}\rangle$	$\langle\frac{474}{17},\frac{6}{17}\rangle$

The fuzzy duration time needs some remarks. It could be considered as uncertain variable which does not depend on the user's decision but in this example, like in [5], it is considered as the controlled variable i.e. decision-maker decides if the duration time is shorter – which is less possible - or it is longer – which is more possible.

The improved schedule was computed with predicate *cons* for the system of all constraints for start times and for all precedences relations between operations.

In Table 2 the improved schedule is presented. The variables $x_1, x_2, \ldots, x_{24}$ indicate that there are many (continuum) improved schedules. CLP(F) module makes use of symbolic computations and finds the relations between values of this variables (15).

$$\begin{cases}
\frac{104}{17} \le x_1 \le 20, & 5 \le x_2, & 4 \le x_4, & x_5 \le 20, \\
\frac{104}{17} \le x_6 \le 20, & 5 \le x_7 \le \frac{104}{17}, & \frac{127}{17} \le x_8 \le 20, & 5 \le x_9 \le \frac{127}{17}, \\
\frac{167}{17} \le x_{10} \le 24, & 5 \le x_{11} \le \frac{46}{3}, & \frac{71}{3} \le x_{12} \le 24, & 5 \le x_{13} \le \frac{71}{3}, \\
9 \le x_{14} \le 24, & 5 \le x_{15} \le \frac{167}{17}, & \frac{309}{17} \le x_{16} \le 24, & 5 \le x_{17}, \\
\frac{104}{17} \le x_{18}, & 1 \le x_{19}, & x_{20} \le 24, & 5 \le x_{21} \le 9, \\
\frac{46}{3} \le x_{22} \le 24, & 5 \le x_{23} \le \frac{309}{17}, & 5 \le x_{24} \le \frac{434}{17}, & x_{18} + x_{19} \le 9, \\
x_3 + x_4 - x_5 \le 0, & x_{18} + x_{19} - x_{20} \le 0, & 0 \le x_3 - x_2, & 0 \le x_{18} - x_{17}, \\
0 \le x_3 - x_{18} - x_{19}. & & &
\end{cases} \tag{15}$$

CLP(F) module prints full information about all improved optimal solutions of considered problem.

6 Conclusions

The presented CLP(F) module for modeling, solving and improving solutions in fuzzy decision-making is consistent with Zadeh's extension principle and Bellman-Zadeh concept of fuzzy decision making. It is implemented with constraint programming in order to fulfill requisite constraints and to deal with a drowning effect related to the sup-min optimization criterion. Proposed module prints in symbolic way full information about all improved optimal solutions. The computational example shows the possibility of improving solutions of fuzzy decision-making with the proposed tool.

The provided module could be used in practical problems. For instance in [10] it was used in two stage heuristic for vehicle routing for clients with fuzzy time windows. In the first stage predicate *cons/2* (two arguments) was used to compute the satisfaction degree for possible routes to find the most satisfied routes and in the second stage predicate *cons/3* (three arguments) was used to compute the improved schedule for routes obtained in the first stage.

The CLP(F) module could be develop in the future research e.g. for support for prioritized constraints [12]. It is also interesting how to implement two different types of descriptors for fuzzy intervals: one for controlled variables (value depends on user decision) and second for uncertain variables (value depends on the nature or some random events).

Acknowledgements

This work was supported in part by the Ministry of Education and Science under grant no. *3T11C05430*.

Bibliography

[1] Richard Bellman and Lofti Zadeh. Decision-making in a fuzzy environment. *Management Science*, 17(4):141–164, 1970.

[2] Rina Dechter. *Constraint processing*. Morgan Kaufmann Publishers, 2003.

[3] Didier Dubois, Helen Fargier, and Henri Prade. *Fuzzy Sets, Neural Networks and Soft Computing*, chapter Propagation and satisfaction of flexible constraints, pages 166–187. Kluwer Academic Publishers, 1994.

[4] Didier Dubois, Hélène Fargier, and Henri Prade. Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. *Applied Intelligence*, 4:287–309, 1996.

[5] Didier Dubois and Philippe Fortemps. Computing improved optimal solution to max-min flexible constraint satisfaction problems. *European Journal of Operational Research*, 118:95–126, 1999.

[6] Didier Dubois and Henri Prade. *Possibility Theory. An Approach to Computerized Processing of Uncertainty*. Plenum Press, 1988.

[7] Christian Holzbaur. OFAI clp(q, r) manual. Technical Report TR-95-09, Austrian Research Institute for Artificial Intelligence, Vienna, 1995.

[8] George J. Klir. Fuzzy arithmetic with requisite constraints. *Fuzzy Sets and Systems*, 91:165–175, 1997.

[9] Przemysław Kobylański. *Soft Computing - Tools, Techniques and Applications*, chapter Improving fuzzy solutions with constraint programming, pages 119–133. Akademicka Oficyna Wydawnicza EXIT, 2004.

[10] Przemysław Kobylański and Michał Kulej. Improved solutions for vehicle routing and scheduling with fuzzy time windows and fuzzy goal. *Badania Operacyjne i Decyzje*, 4, 2003.

[11] Przemysław Kobylański and Paweł Zieliński. Fuzzy modeling with constraint technology. In *Proceedings of EUROFUSE 2002, 7-th Meeting of the EURO Working Group on Fuzzy Sets, Workshop on Information Systems*, 2002.

[12] Xudong Luo, Jimmy Ho-man Lee, Ho-fung Leung, and Nicholas R. Jennings. Prioritised fuzzy constraint satisfaction problems: axioms, instantiation and validation. *Fuzzy Sets and Systems*, 136:151–188, 2003.

[13] H.T. Nguyen. A note on the extension principle for fuzzy sets. *J. Math. Anal. Appl.*, 68:369–380, 1978.

[14] Ulf Nilsson and Jan Maluszynski. *Logic, Programming and Prolog.* John Wiley & Sons Ltd., 1995.

[15] L. A. Zadeh. *The concept of a linguistic variable and its application to approximate reasoning.* American Elsevier Publishing Co., 1973.