Cooperative Control achieved by Generic Genetic Fuzzy Logic

Octavian Cuibus , Tiberiu Leţia

Technical University of Cluj-Napoca, Cluj-Napoca, Romania,
(e-mail: octavian.cuibus@ aut.utcluj.ro, tiberiu.letia@aut.utcluj.ro)

Abstract: The paper presents three cooperative control strategies with application to the problem of lake system control. The control system is implemented by a distributed set of communicating agents that use fuzzy logic rules to compute the control decisions. In order to improve the global control performance, each agent sends the control context to its neighbours. The information included in the context and the manner the context exchange is made establish the control strategy: implicit cooperation, coordination and explicit cooperation. Improving the control performance involves more terms in premises of the fuzzy logic control rules, increasing the number of logic rules. Hierarchical fuzzy architectures are used to diminish the number of rules for different methods. The unknown fuzzy logic control rules are obtained offline by the generic genetic approach. The simulations results are given for the three cooperative control strategies and a performance comparison is made.

Keywords: distributed control, cooperative control, generic genetic algorithms, hierarchical fuzzy control.

1. INTRODUCTION

The cooperative control strategy involves several agents that perform shared tasks. These tasks are based on the relationship between the contexts of individual agents. The decisions and the behavior of each agent influence the contexts of other agents so they have to act according to a consensus. The cooperative control captures those problem areas which imply some type of repetition of identical or non-identical interconnected subsystems (Khatir et al. (2003)).

Cooperative control can successfully approach decentralization since it allows the development of complex behavior based on several combined controllers in order to achieve the desired result (Innocenti et al. (2007)).

The centralized control methods use one actor to control the whole system, whereas the distributed approach splits the system into several controlled subunits, each with its own controller. The main advantage is that when a controller fails or its communication is interrupted the rest of the system can continue to behave correctly. A subsystem with a failed controller or failed communication can still influence other subsystems. In this case, the influence of the subsystem that quits the cooperation is regarded as a disturbance and managed accordingly.

Since each controller is responsible only for its assigned subsystem, the global system solution may not be optimal since it is composed of the local solutions. According to the cooperative systems strategy, the agents take into account the other agent behaviors or the control context with the intention of improving their own control performances.

Generally, the scale of the system prevents from finding of an optimal solution, but it is possible to obtain a suboptimal one, close to the best behavior. This involves improving the local control decisions to fulfill the global requirements.

The main advantages of cooperative control are as follows:

- computational load decrease or distribution
- control performance increase
- increase of system reliability and flexibility
- building a portable system that may be implemented for a large application domain

One of the goals of the current approach is to obtain few and simple fuzzy control rules that can be implemented to fulfill some real-time constraints. Control rule construction can be performed offline, therefore methods that require large computational effort may be used.

This paper is a logical extension of an earlier paper, “Implicit cooperative control achieved by generic genetic fuzzy logic”, (Leţia et al. (2010)), in which only the implicit cooperative strategy is thoroughly presented. The cooperative control structures have been presented in (Leţia et al. 2010), but only as a theoretical concept. As the main contribution, this paper introduces new various methods to implement the cooperative control strategies: implicit, coordinated and explicit cooperation. Of course, the independent and implicit strategies implementation from the above mentioned paper are put forward once again for performance comparison.

2. RELATED WORKS

Several studies present various distributed control systems, including multi-agent approaches. The focus is set on the fuzzy approach and various cooperation strategies. Some of the relevant papers are cited below.

(Korotkich (2002)) considers a collection of many subsystems that solve an optimization problem by cooperating with each other in order to find control actions and criteria such that the whole system maximizes a given performance function.

(Waldock et al. (2008)) point out two cooperative strategies: implicit cooperation, based on the idea of building and maintaining a common distributed picture (which is referred to as "context" in this paper), and explicit cooperation, that implies negotiation in order to agree on a common distributed plan. The explicit cooperation is achieved by exploiting factorization in the utility function to form a factor graph.

(Hakimi-Asiabar et al. (2009)) state that a multi-reservoir operation can be regarded as a multi-objective problem with nonlinear, non-convex and multi-modal objective functions. Genetic algorithms are used because of their feature of probabilistic search which makes them capable of solving a variety of complex multi-objective optimization problems, including non-linear, non-convex and multi-modal functions.

(Innocenti et al. (2007)) develop control architectures for distributed independent asynchronous behavior of multi-agent systems. The cooperative control allows the development of complex behavior based on several controllers combined to achieve the desired result.

The main problems that should be approached when dealing with Cooperative Control of Distributed Multi-Agent Systems are, as stated by (Shama (2007)) in his book: distributed control and computation, adversarial interactions, uncertain evolution and complexity management.

(Hong and Chen (2000)) propose two fuzzy learning methods for automatically deriving membership functions and fuzzy *if-then* rules from a set of given training examples. Appropriate initial membership functions are built by selecting relevant attributes and then simplifying the intervals and membership functions of each attribute to form a decision table. The fuzzy strategy is used for water flow and water level control for one basin by (Nezam et al. (2002)).

(Chiou and Lan (2005)) concluded that logic rules and membership functions are two key components of a fuzzy logic controller. If both components are calculated simultaneously using genetic algorithms, a very long chromosome is needed, which may deteriorate performance. They propose a bi-level iterative evolution algorithm in selecting the logic rules and tuning the membership functions. (Buiu et al. (2003)) also use genetic fuzzy logic control, but for a different purpose.

A new architecture for cooperative control implementation is proposed by (How et al. (2009)).

A class of hierarchical fuzzy system with constraints on the fuzzy rules is conceived by (Hoffman et al. (1994) and Joo et al. (2007)), in order to diminish the fuzzy logic rule number. Other rule compression techniques are used successfully by (Gegov et al. (2007) and Koczy (1997)), employing simple ellimination of redundant rules and respectively using interpolation of elliminated rules to construct the lost rules with a certain pre-established precision. In this paper, rule compression is done by hierarchization of fuzzy blocks so that the number of decision inputs remains the same, but the number of fuzzy rules is reduced.

(Klavins et al. (2004)) use the Computation and Control Language to construct tools for cooperative control systems.

(Choi et al. (2009)) use more complex methods for cooperation implementation, mainly consensus-based auction algorithms. These utilize a market-based decision strategy as the mechanism for task selection and use a consensus routine based on local communication as the conflict resolution mechanism to achieve agreement on the winning bid values.

3. THE LAKE SYSTEM MODEL

The cooperative control strategy is applied to a lake system such as the one presented in Figure 1. Such lake systems are used to supply water for human residences, farms or irrigation systems. The control goal is to maintain the water levels close to the specified (reference) values, despite the variable disturbances that may affect the system, such as environment conditions (rain, evaporation), or variable flow demands at the output of the downstream lake.

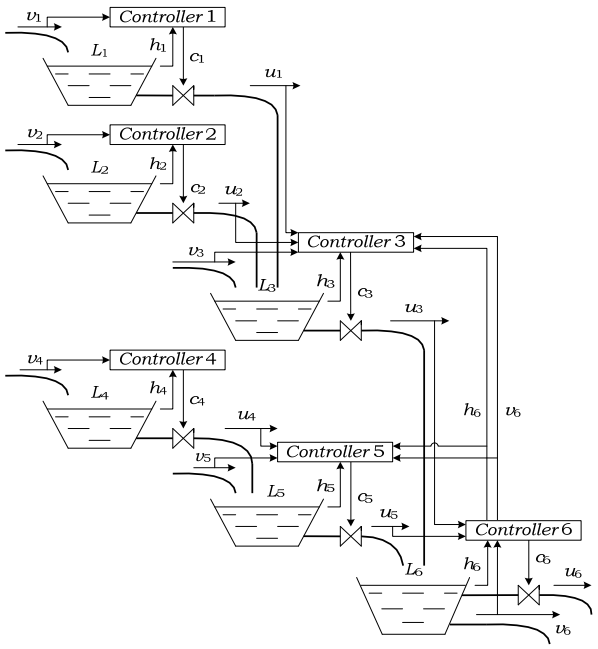

Fig. 1. Lake system.

Each controller assigned to a lake can measure the water level and estimate the disturbance input flow, but not from the upstream lakes. The agents implementing the controllers are able to communicate at least with a subset of other agents (the neighbouring agents). All agents can interchange information before they make their own decisions.

In Figure 1, the data for lake L_i (i=1, ..., 6) is denoted by:

- v_i the uncontrolled input flow from the environment (disturbance measured in m^3/s);
- h_i the current water level (measured in m);
- c_i the control signal for the output flow;
- u_i the controlled output flow (measured in m^3/s);

The discrete time model of one lake is presented in detail in the paper "Implicit cooperative control achieved by generic

genetic fuzzy logic", (Leția et al. (2010)) and only stated below for lake *i*, for reference.

$$x_i(k+1) = x_i(k) + \frac{u_{prev}(k) + v_i(k) - u_i(k)}{S_i^0 + g_i\left(r_i + x_i(k)\right)\cdot\left(r_i + x_i(k)\right)} \quad (1)$$

In formula (1), the following notations were used:

- r_i is the reference level of lake *i*
- $x_i(k) = r_i - h_i(k)$ is the level error at time *k*
- $v_i(k)$ is the disturbance input flow
- $u_i(k)$ is the controlled output flow
- *prev* denotes the lakes located upstream of lake *i*
- S_i^0 and g_i represent the base surface and the shape of the basin walls

The lake system model is composed of six equations similar to (1), considering the links between lakes. The steady state gives the nominal values of the variables v_i and u_i.

4. CONTROL SYSTEM PHILOSOPHY

The control problem is to calculate the value $u_i(k)$ such that the level error $x_i(k)$ is cancelled in the minimum possible time and remains as small as possible at steady state.

As an extension of the cooperative control categories presented by (Waldock et al. (2008)) in their paper, three cooperative control strategies are implemented in this paper:

- implicit cooperative control,
- coordinated cooperation and
- explicit cooperative control.

The difference between these methods consists of the information used for calculating the control actions and whether an agreement is reached in the calculation process.

The agents share context information by means of a communication system. They perform shared tasks, where the tasks depend on the relationship between the contexts of individual agents. The control and cooperative functions are performed by the agents associated to each lake.

4.1. Agent structure

Unlike the general multiagent architecture (for example, the one presented by (Innocenti et al. (2007)), the proposed architecture uses identical control units (agents), where each agent is usually composed of:

- sensing components (sensors),
- communication component and
- control component.

Each agent has a *Context Constructor* component that builds the control context using the measured or communicated information. The agent control context is composed of local state (process measurements) plus other information, such as:

- constraints,
- measured or estimated disturbances, and their predictions (disturbance forecasts),
- operator input (setpoints, requirements or other recommendations),
- inferenced and statistical information from the past,
- information communicated by other agents (state measurements, control decisions or intentions, etc.) and
- the intended control action for the next step.

The type of information included in the context and used for the calculation of the control action determines the cooperative control strategy.

The key contribution of this component is the replacement of state-based control by the context-based control.

4.2. The control task

The agents perform the shared task of minimizing the global performance function (2). The problem is in fact a multi-objective problem (Hakimi-Asiabar (2009)), but here all the objectives are integrated into a single function (2).

$$J = \sqrt{\sum_{i=1}^{N}\sum_{k=0}^{K} p_i \cdot \left(x_i(k)\right)^2} \quad (2)$$

In formula (2), p_i is the priority of each lake, N corresponds to the number of lakes and K to the time control horizon. Thus, a performant controller should minimize the level total error, despite the disturbances (v_i) that may occur.

Due to the distribution, only local levels can be directly measured by the agents, but the calculated control commands have to take into account the other agent behaviors (context information) as well, as described below.

4.3. Control system architecture

The control structure for *implicit cooperative control* strategy can be seen in Figure 2. The agents communicate local context information and then calculate the control actions in one single step. No agent is thus aware of the control decision of any other agent.

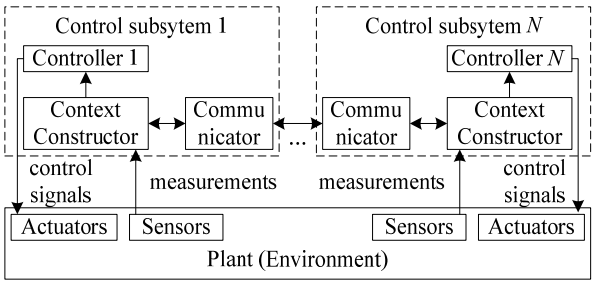

Fig. 2. Implicit cooperative control.

The *coordinated cooperative control* structure is presented in Figure 3. Unlike the case of implicit cooperative control, the exchanged information can include controller decisions, not

just context information. Apart from being a simple intermediate of information exchange, the coordinator can also apply logic rules to process data that is to be transmitted to each controller (agent), in a unitary way, as presented by (How (2009)) for his cooperative control architecture.

The chronology of activities in one sample period is as follows: the context constructor takes process measurements and sends this information to the controller. Each controller calculates the information that is to be sent to the coordinator (which may include the intended control actions, but not the final control actions). The coordinator processes data from all controllers and provides the results to each controller. Based on this new information, each controller calculates the final control action, which is applied to the process.

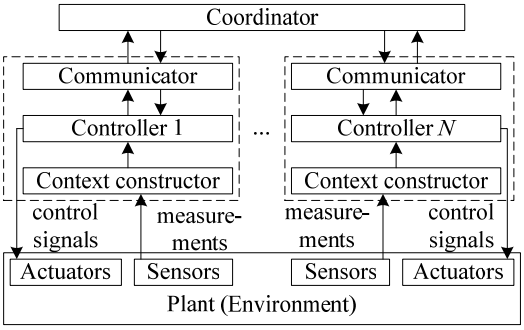

Fig. 3. Coordinated cooperative control.

The system layout for *explicit cooperative control* can be seen in Figure 4. This is the most specialized type of cooperative control and implies explicit communication and cooperation between agents in an unspeciffied number of steps (iterations).

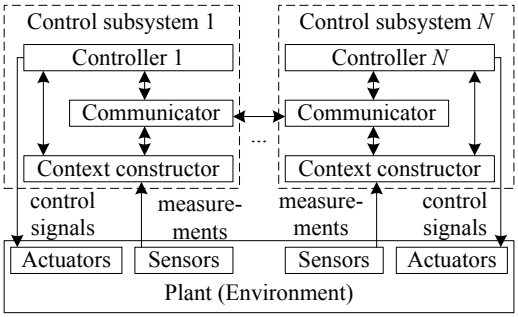

Fig. 4. Explicit cooperative control.

The context constructor first builds the initial context by taking process measurements and other possible external information (from the operator, etc). The context can be then sent directly to other controllers or can be first processed in a certain way. The controller calculates the intended control action and sends it to other controllers. In response, they communicate their new option regarding this or their own control action. An iterative message exchange process takes place at which end the controllers reach an agreement about what control action should be used at the present moment.

Message exchange can imply controller options, considering how much each controller gives up as utility and how much it accepts as cost. Powerful strategies can be used here, including game theory strategies. A major concern to look after is avoiding infinite loops in the cooperation process.

4.4. The fuzzy approach

In all strategies, fuzzy logic is used to build the control action (Leţia et al. (2010)). When constructing the context, each agent calculates the fuzzy logic values of the variables $x_i(k)$ and $v_i(k)$, which are denoted by x_i and v_i respectively. The mentioned fuzzy logic variables take values in the discrete universe $\{H^-, L^-, Z, L^+, H^+\}$ (with H, L, Z meaning high, low and zero). Each agent calculates the control action u_i as fuzzy value using one or more IF…THEN… inferences, according to the employed strategy.

The membership function of x_i is presented in Figure 5 and is similar to the membership functions presented by (Nezam et al. (2002)). Standard triangular shaped membership functions are used for v_i and u_i. The fuzzy values are centered around the nominal values of the variables, namely $x_i^0 = r_i$, v_i^0, u_i^0. The discrete universe for u_i is $\{E^-, H^-, M^-, L^-, Z, L^+, M^+, H^+, E^+\}$ (with E meaning extreme high and M medium). All membership functions have been established apriori.

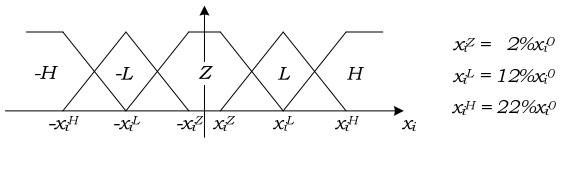

Fig. 5. Water level membership functions.

4.5. The generic genetic algorithm

The fuzzy logic control sets are calculated offline by employing generic genetic algorithms, using the set of all possible combinations of disturbances. This approach is similar to using only the upper level of the genetic fuzzy controller proposed by (Chiou et al (2005)). Another set of relevant disturbances was used to test the system behavior and evaluate the control performances.

The genotype is set according to the fuzzy rules used by the employed strategy. The length of the chromosome is equal to the number of fuzzy rules required by the control strategy and the genes correspond to the elements of the rule matrix (see Leţia et al. (2010)) The alleles are chosen from the elements of the discrete universe for u_i, unless specified otherwise.

When evaluating each chromosome, the system must be simulated starting from each possible initial (fuzzy) state, such that the discrete universe is entirely covered, using each possible set of disturbances. The simulation is performed by running the system composed of 6 equations of the form (1) in a loop spanning across the time horizon K. The performance of the control strategy is evaluated using the fitness function of the form (2).

5. INDEPENDENT CONTROLLERS

As a standard for control performance, independent control is also considered. An independent controller takes into account only the local error (state) x_i and the measured/estimated disturbance v_i. The fuzzy logic formula is:

$$IF\ (x_i = X_i) \wedge (v_i = V_i)\ THEN\ (u_i = U_i) \tag{3}$$

where X_i and V_i take values in the discrete universe.

Formula (3) leads to 25 logic rules for one lake, which can be arranged in matrix form (Leţia et al. (2010)). Note that lakes {1, …, 5} can be controlled using the same logic rules, but another set of rules must be used for lake 6 since v_6 is an output flow. The fuzzy logic control sets are constructed offline using the generic genetic algorithm described above.

6. IMPLICIT COOPERATIVE CONTROL

The implicit cooperative control structure is represented in Figure 2. This is the case when the agents send each other information about the error, disturbance or previous control decisions, but not about the current control actions. The reason is that the previous control decisions $u_i(k)$ are available, but not the next $u_i(k+1)$ control action. The sending of the previous control action to neighbors can improve the estimation of the disturbances and also the control performances.

Each agent sends the coordination vector $[x_i, v_i]$, and then, having received the equivalent information from other agents, calculates the output command with the fuzzy logic formula:

$$\begin{aligned} &IF\ (x_1 = X_1) \wedge \ldots \wedge (x_N = X_N) \wedge \\ &\quad (v_1 = V_1) \wedge \ldots \wedge (v_N = V_N)\ THEN\ (u_i = U_i) \end{aligned} \tag{4}$$

where X_i and V_i take values in the discrete universe.

The number of necessary fuzzy logic rules of the form (4) is $5^N \cdot 5^N = 5^{12}$. Due to the huge number of rules, the formula (4) is not practically applicable, thus the number of rules must be reduced to a reasonable level by suitable methods.

Method 1: The controller of the lake L_i takes into account the state x_i, the disturbance v_i and the state of another neighbor lake L_j considered strongly influenced by the control decisions taken for the lake L_i. For this problem, L_j is considered the downstream lake for lake L_i.The fuzzy logic control rules are of the form:

$$IF\ (x_i = X_i) \wedge (v_i = V_i) \wedge (x_j = X_j)\ THEN\ (u_i = U_i) \tag{5}$$

To avoid the effect of dimensionality a hierarchical structure was used (Joo et al. 2007), which is presented in Figure 6.

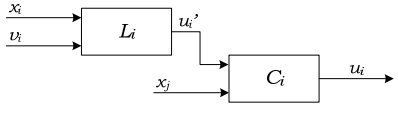

Fig. 6. Hierarchical fuzzy structure for implicit control (method 1)

The number of necessary rules is 25 (for block L_i) and 45 (for the block C_i). They have the form:

$$\begin{aligned} &IF\ (x_i = X_i) \wedge (v_i = V_i)\ THEN (u_i' = U_i) \\ &IF\ (u_i' = U_i) \wedge (x_j = X_j)\ THEN (u_i = U_i) \end{aligned} \tag{6}$$

The controller of Lake 6 is the same as for the independent control, since Lake 6 has no downstream lake.

Method 2: The controller of lake L_i improves its control decisions using the disturbance of lake L_j. The rules are:

$$\begin{aligned} &IF\ (x_i = X_i) \wedge (v_i = V_i) \wedge \\ &\quad (x_j = X_j) \wedge (v_j = V_j)\ THEN\ (u_i = U_i) \end{aligned} \tag{7}$$

As in the previous case, the number of rules is diminished by a hierarchical structure, which is presented in Figure 7.

The blocks R_i, R_j and C_i implement fuzzy rules with the form according to their inputs and outputs. The necessary number of control rules is 25 for block R_i, 25 for R_j, and 81 for the block C_i.

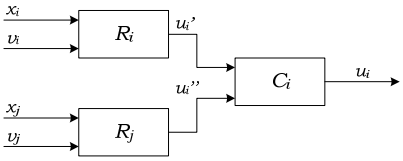

Fig. 7. Hierarchical fuzzy structure for implicit control (method 2)

As with Method 1, the controller of Lake 6 is in fact an independent controller, since there is no downstream lake.

Method 3: The construction of controllers that use the errors of three neighbor lakes and their own level error is based on the formula (8), but hierarchical structures can also be used.

$$\begin{aligned} &IF\ (x_i = X_i) \wedge (x_j = X_j) \wedge \\ &\quad (x_k = X_k) \wedge (x_l = X_l)\ THEN\ (u_i = U_i) \end{aligned} \tag{8}$$

Using similar structures many other control algorithms can be constructed. The chosen input variables and their number affect the control performances.

7. COORDINATED COOPERATIVE CONTROL

The coordinated cooperative control structure is represented in Figure 3. This is the case when the agents can send information about state and disturbance or even previous control decisions, but not about the current control actions. The reason for this is that, during a certain sampling period, the previous control decisions $u_i(k)$ are available, but the next control actions $u_i(k+1)$ are not. Nevertheless, the controller can also send information about the intended control actions. Sending of the previous and/or intended control actions to the coordinator and then to the neighbors can improve disturbance estimation and control performance.

The chronology of the events that take place in one time step has been described in subsection 4.3. The implementation of

the coordinated cooperative control using fuzzy control blocks is described below.

First, each controller calculates the information a_i that is sent to the coordinator, based on the information available in the local context. The coordinator takes the fuzzy values a_i from each cotroller and calculates the coordination vector $[b_i]$ using the fuzzy logic rules:

$$IF\ (a_1 = A_1) \wedge \ldots \wedge (a_n = A_n)$$
$$THEN\ (b_1 = B_1); \ldots; (b_n = B_n) \quad (9)$$

Each coordination value b_i is sent to the appropriate controller. The controllers then calculate the control actions, based on the local error (state) x_i, the disturbance v_i, and the new information b_i, with the fuzzy logic rule:

$$IF\ (x_i = X_i) \wedge (v_i = V_i) \wedge (b_i = B_i) THEN\ (u_i = U_i) \quad (10)$$

On the general case, a_i and b_i can also be vectors. Depending on the implementation, the a_i may include information about the disturbance, water level, previous or intended control action for lake i, or even the way that controller i wants the upstream controllers to behave, by setting upper and lower bounds on the outputs of the upstream lakes. These bounds can be sent by means of the vector a_i. Similarly, b_i can represent processed information that is to be sent to controller i about the state of the downstream lakes, the recommeded output of the upstream lakes, etc. Note that a_i and b_i may also contain different types of data for different controllers.

If 5 fuzzy levels are considered for a_i and b_i, the number of necessary fuzzy logic rules of the form (9) for the coordinator is $N \cdot 5^N = 6 \cdot 5^6$ and of the form (10) for each controller is $5^3 = 125$. As for implicit control, the use of fuzzy inferences (9) and (10) is not possible due to the huge number of rules needed. This requires some diminishing con-structions, to reduce this number down to a reasonable level.

Method 1: For this particular method, it is considered that each value a_i is actually a vector that represents the water quantity in the lake and has two components: x_i and v_i.

In order to reduce the number of rules for the coordinator, some pre-processing of the values a_i must be performed. The values x_g and v_g, are calculated with the formulas (11), where x_i and v_i are the components of the vector a_i.

$$x_g = \sum_i x_i \cdot \alpha_i$$
$$v_g = \sum_i v_i \cdot \beta_i \quad (11)$$

The quantitiy x_g represents approximatively the total water quantity accumulated in the system, and v_g the water quantity that is about to enter in the system. The coefficient α_i represents the volume of lake i, at the nominal water level, relative to the total volume of the lake system, and β_i is the reciprocal of the surface of lake i, at the nominal water level. The reason for taking the reciprocal of the surface is that v_g actually represents the expected rise of the water level because of input v_i. Also, since v_6 is an output flow, β_6 is considered negative.

The membership functions for x_g and v_g are obtained by substituting the fuzzy levels of x_i and v_i into formulas (11).

The structure in Figure 8 is thus used for the coordinator, and the coordinator fuzzy logic control rules from (9) are now of the form (12).

$$IF\ (x_g = X_g) \wedge (v_g = V_g)\ THEN\ (b_g = B_g) \quad (12)$$

The quantities b_i are replaced by a single parameter, b_g, which contains information about the distribution of water in the system. The membership functions for b_g are not important, since b_g is an input parameter in the controller structure (Figure 9) and is never defuzziffied.

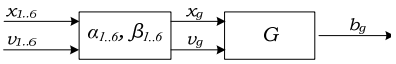

Fig. 8. Coordinator structure (coordinated control, method 1)

The number of controller fuzzy logic rules (10) is diminished using the structure from Figure 9 (see Joo et al. 2007), where each block implements appropriate fuzzy inferences.

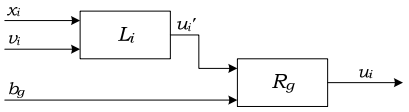

Fig. 9. The hierarchical structure for each controller (coordinated control, method 1)

The necessary number of control rules is 25 for block G (for the coordinator), 25 for block L_i and 81 for block R_g (for each controller). Since b_g and u_i' are variables on 9 fuzzy levels, the block R_g must have 81 logic rules. In order to reduce this number further, it can be considered that b_g and u_i are first defuzzyfied according to the 9 fuzzy levels and re-fuzzyfied on 5 new fuzzy levels: $\{H^-, L^-, Z, L^+, H^+\}$. The membership functions are standard and tuned accordingly: the value of u_i^H must be identical in the two fuzzy universes. Block R_g has thus only 25 logic rules.

Note that lakes $\{1, \ldots, 5\}$ can be controlled using the same logic rules, but another set of rules must be used for lake 6.

Method 2: Each controller first calculates the quantity a_i as the intended control action u_i', using the independent control equation (3). The quantity a_i that is sent to the coordinator is a vector that contains u_i' and the water level x_i.

The coordinator then calculates for each controller a recommended control action u_i'', considering the intended control actions of each controller and its effect on the levels of other lakes. The coordinator implements formula (13),

$$IF\ (u_i' = U_i) \wedge (x_j = X_j)\ THEN\ (u_i'' = U_i) \quad (13)$$

where j is the lake that has the highest sensibility to the output of lake i (i.e. the lake that is downstream from i). Note that the form of the premise for the logic rules is different for each u_i'' that is to be calculated. The recommended output u_i'' is then sent to each controller as b_i.

Using the recommandation u_i'', each controller calculates the control action u_i using the formula (14), which can be also organized in a hierarchical structure (Figure 10) in order to reduce the number of logic rules.

$$IF\ (x_i = X_i) \wedge (v_i = V_i) \wedge (u_i'' = U_i)\ THEN\ (u_i = U_i) \tag{14}$$

The number of new logic rules needed to implement the controllers using this method is 25 for block S_i and 81 for the block U_i, which can be reduced to 25 using the same defuzzyfication and re-fuzzyfication strategy as for method 1.

Lakes {1, ..., 5} can be controlled using the same logic rules, whereas lake 6 uses the the independent control strategy, since the output of lake 6 does not affect any other lake.

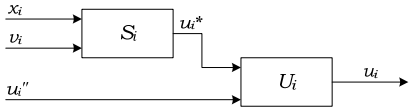

Fig. 10. Reduced controller structure (coordinated control, method 2)

The number of logic rules for the coordinator (equation (13)) is 45 and can be reduced to 25 using the previously described strategy. The same rules can be used in relation to each lake, but better precision may be achieved using custom rules for each lake.

8. EXPLICIT COOPERATIVE CONTROL

The explicit cooperative control strategy presented in Figure 4 is representative for the idea of cooperative control. According to this strategy, the control actions $u_i(k)$, are calculated for each time sample k in an iterative process in many steps, using information from all agents. Agents cooperate and negotiate over what control action they should apply to the process in the next time step. The process of negotiating a speciffic value involves explicit communication and cooperation between the agents that are interested in that value. To a certain extent, the explicit cooperative strategy is similar to the coordinated strategy with the remark that here, each agent has some coordinator behavior included. The coordinator is thus distributed into each individual controller.

The chronology of the steps by which the controllers calculate the control actions is presented below. The variable a_i denotes the coordination value that controller i calculates and distributes to each interested controller. For each step, the general fuzzy logic implementation is given below for the lake control system problem.

- *Step 1*: each controller i calculates a_i and sends it to the appropriate (interested) controllers:

$$IF\ (x_i = X_i) \wedge (v_i = V_i)\ THEN\ (a_i = A_i) \tag{15}$$

- *Step 2*: based on the information a_j, each controller i updates the value of a_i and sends it again to other controllers.

$$IF\ (x_i = X_i) \wedge (v_i = V_i) \wedge (a_1 = A_1) \wedge \ldots \wedge (a_n = A_n)\ THEN\ (a_i^{new} = A_i) \tag{16}$$

Step 2 is repeated until the difference between a_i and its updated value is small enough (see condition (17), ε being a small number which measures performance).

$$\left| a_i - a_i^{new} \right| < \varepsilon \tag{17}$$

Before the next iteration, the initialization $a_i = a_i^{new}$ must be made.

- *Step 3*: based on the information a_j (j = 1..6), each controller calculates the control action u_i (equation (18)).

$$IF\ (x_i = X_i) \wedge (v_i = V_i) \wedge (a_1 = A_1) \wedge \ldots \wedge (a_n = A_n)\ THEN\ (u_i = U_i) \tag{18}$$

Typically, a_i contains information about the intended control action, but on the general case, a_i can be a vector containing upper and lower bounds for the control action, critical signals or information that other controllers must be aware of, etc. As an extension, it can be considered that not all controllers see the same information about the unit i.

As already mentioned, special care must be taken to ensure that the algorithm does not reach infinite loop (i.e. condition (17) is validated at certain iteration). One of the possible strategies involves genetic algorithms, solving the convergence problem by elliminating individuals that reach infinite loop. Because infinite loop might not always be correctly detected, a possible approach is to elliminate the individual that has not fulfilled condition (17) in a maximum of 10 to 20 iterations. This should not be regarded as a performance measure but rather as an elimination criterion.

If we consider that a_i is the intended control action (on 9 fuzzy levels), then implementing the control problem with fuzzy logic rules of the form (15), (16) and (18) requires $25+25\cdot9^N+25\cdot9^N = 25+50\cdot9^6$ fuzzy rules. As for the other strategies, this number must be diminished to a reasonable level for implementation, using the methods presented below.

Method 1: Instead of using an intermediate coordination variable a_i in the logic rules, the intended control action u_i can be used, thus elliminating step 3.

Moreover, for lake i, not all information from all lakes is relevant, but only that related to the immediately downstream and upstream lakes, denoted ↓ and ↑. The steps of the explicit cooperative control strategy, for lake i, are rewritten below.

- *Step 1*: each controller i calculates the intended control action u_i according to the independent strategy (3) and sends it to the lakes denoted by ↓ and ↑.

- *Step 2*: Based on the information about the desired control action of the upstream and downstream lakes, each lake i calculates the new intended control action using formula (19). At this step, lake i should "observe" if the controller of the downstream lake is trying to fill or empty the lake and

help it in the process, if possible, by delivering controlled output to the lake downstream. The desired output of the upstream lake is used in order to better approximate the total input in lake *i*.

$$IF\ (x_i = X_i) \wedge (v_i = V_i) \wedge (u_\uparrow = U_\uparrow) \wedge (u_\downarrow = U_\downarrow)$$
$$THEN\ (u_i^{new} = U_i) \tag{19}$$

Step 2 is repeated until the condition equivalent to (17) is validated for each lake *i*, for $\varepsilon = 5\% \cdot r_i$. Step 3 being eliminated, the control action of lake *i* is the last value of u_i^{new}.

In order to reduce the number of logic rules, formula (19) can be organized hierarchically as shown by Figure 11. The quantity $v_i+u_\uparrow$ is the total input in lake *i* and is evaluated in the discrete universe U_i since it is equal to total output at nominal values. If there are two lakes directly upstream of lake *i*, then the sum of the two outputs is considered when calculating $u_\uparrow$. If there is no upstream lake (as it is the case for lakes 1, 2 and 4), $u_\uparrow$ is simply omitted.

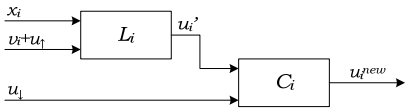

Fig. 11. Controller structure – explicit cooperative control, method 1, step 2

Using this method, the total number of new necessary logic rules is reduced to 45+81 = 126 for each lake. Lake 6 has no downstream lake and it is controlled by using the same strategy, but without using block C_i in step 2 (u_6' has the role of u_6^{new}). Moreover, the quantity v_6 is always taken as negative, since v_6 is an output.

The drawback of this method is that the communication between controllers only takes place in an informative way. No controller can send information about its option regarding the output of other controllers.

Method 2: Each controller communicates information about its intended control action to the downstream lake, so it can adjust its own control action to cope with the given input, or propose new intended control action for the upstream lake. Using this method, the variable a_i is a vector that contains the lower and upper bounds for the control action. The idea is to reduce the interval within the bounds at each iteration. Agent *i* communicates with the upstream and downstream agent.

Denote by $a_{i,\min}^j$ and $a_{i,\max}^j$ the lower and upper bounds for the control action of lake *i*, as calculated by lake *j*. Most often, *j*=*i*, but in special situations a downstream lake can propose bounds for the output of the upstream lake. Also, notations ↑ and ↓ are used. Below is a rewriting of the steps of the explicit cooperative control strategy, for lake *i*.

- *Step 1*: each controller *i* calculates the interval in which its own control action should be in order to cope with its inputs and water level. The upper and lower limits are calculated according to formula (20). Note that the discrete universe A_i has 9 fuzzy levels, since it refers to bounds on the output flow u_i.

$$IF\ (x_i = X_i) \wedge (v_i = V_i)\ THEN\ (a_{i,\min}^i = A_i); (a_{i,\max}^i = A_i) \tag{20}$$

- *Step 2*: Based on the information regarding the control action bounds of the upstream lake, each controller *i* updates its own control action bounds. If controller *i* cannot cope with the water flow from the upstream lake, given by $a_{\uparrow,\min}^{\uparrow}$ and $a_{\uparrow,\max}^{\uparrow}$, it might want to modify these bounds, calculating $a_{\uparrow,\min}^i$ and $a_{\uparrow,\max}^i$. These latter quantities are thus included in the conclusion of inference (21). Of course, the first strategy should be to modify its own bounds and then propose new bounds for the upstream controller.

Since a controller is not the only one responsible for modifying its control action bounds (the downstream controller might also do this, by $a_{i,\min}^{\downarrow}$ and $a_{i,\max}^{\downarrow}$), it is mandatory that these bounds are included in the premise of the formula (21).

$$IF\ (x_i = X_i) \wedge (v_i = V_i) \wedge (a_{\uparrow,\min}^{\uparrow} = A_\uparrow) \wedge$$
$$(a_{\uparrow,\max}^{\uparrow} = A_\uparrow) \wedge (a_{i,\min}^{\downarrow} = A_i) \wedge (a_{i,\max}^{\downarrow} = A_i) \tag{21}$$
$$THEN\ (a_{i,\min}^i = A_i); (a_{i,\max}^i = A_i); (a_{\uparrow,\min}^i = A_\uparrow); (a_{\uparrow,\max}^i = A_\uparrow)$$

If at certain iteration, the upstream controller calculates its own control action interval and also controller *i* establishes new bounds for the controller upstream, the interval taken for the next iteration is the intersection of the two intervals.

Step 2 is repeated until condition (22) and the equivalent *min* form are validated for each lake *i*. The variable *k* represents the iteration.

$$\left| a_i^{\max}(k+1) - a_i^{\max}(k) \right| < \varepsilon \tag{22}$$

Another possible condition for ending the loop is (23). In both cases, ε can be taken as $\varepsilon = 5\% \cdot r_i$.

$$\left| a_i^{\max} - a_i^{\min} \right| < \varepsilon \tag{23}$$

- *Step 3*: based on the control context and the last calculated lower and upper bounds for the control action of the upstream lake, each controller calculates the exact value of its control action u_i*, according to formula (24) and Figure 12a). The value of the control action is then trimmed according to (42) by the block T_i from Figure 12a) to the previously calculated interval $\left[a_{i,\min}^i, a_{i,\max}^i\right]$, resulting u_i.

$$IF\ (x_i = X_i) \wedge (v_i = V_i) \wedge (a_{\uparrow,\min}^{\uparrow} = A_\uparrow)$$
$$\wedge (a_{\uparrow,\max}^{\uparrow} = A_\uparrow)\ THEN\ (u_i{*} = U_i) \tag{24}$$

$$u_i = \min(u_i{*}, a_i^{\max})$$
$$u_i = \max(u_i{*}, a_i^{\min}) \tag{25}$$

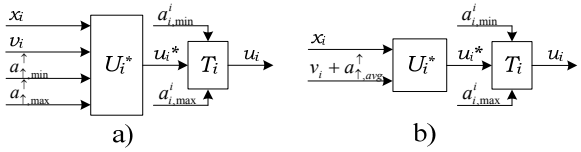

Fig. 12. Controller structure - explicit cooperative control, method 2, step 3. a) general controller structure, b) reduced controller structure

A simplifyed structure must be used for lakes that are the most upstream or the most downstream. For lakes 1,2 and 4, the quantities $a^{\uparrow}_{\uparrow,\min}$, $a^{\uparrow}_{\uparrow,\max}$, $a^{i}_{\uparrow,\min}$, $a^{i}_{\uparrow,\max}$ should be omitted from the structure. For lake 6, $a^{\downarrow}_{i,\min}$ and $a^{\downarrow}_{i,\max}$ should be omitted.

The structure of the algorithm as presented here requires a total number of $2\cdot5^2+4\cdot5^2\cdot9^4+5^2\cdot9^2 = 658.418$ logic rules for each lake. The use of genetic algorithms to find the control rules is excluded due to the large number of rules. Some diminishing constructions must be used, as described below.

Instead of using the fuzzy inference from (21), the lower and upper bounds of the control action could be calculated separately. The reduced structure for the lower bound (Figure 13) uses the fuzzy block A_i^{min} to reduce the number of rules by considering the lower limit on the total input flow in lake i, $v_i + a^{\uparrow}_{\uparrow,\min}$. The block A_i^{min} implements the fuzzy inference (26). Instead of calculating $a^{i}_{\uparrow,\min}$ directly, one output of the block A_i^{min} represents the total minimum input flow that lake i can accept and is denoted $(v_i + a^{i}_{\uparrow})^{*}_{\min}$. Based on the outputs of block A_i^{min}, the allowed lower limit for the output of the upstream lake $a^{i}_{\uparrow,\min}$ is then calculated in block T_i, by subtracting the known input v_i, as shown in (27). Similar procedure is used for calculating the upper limit of a_i.

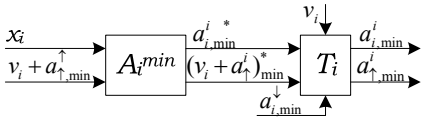

Fig. 13. Reduced controller structure for the lower bound of a_i (similar for upper bound) – explicit cooperative control, method 2, step 2

$$IF\ (x_i = X_i) \wedge (v_i + a^{\uparrow}_{\uparrow,\min} = U_i)$$
$$THEN\ (a^{i\ *}_{i,\min} = A_i); ((v_i + a^{i}_{\uparrow})^{*}_{\min} = U_i) \quad (26)$$

The limits on the control action bounds, $a^{i\ *}_{i,\min}$ and $a^{i\ *}_{i,\max}$ must be then trimmed to the interval $\left[a^{\downarrow}_{i,\min}, a^{\downarrow}_{i,\max}\right]$ by the block T_i, according to (27) its equivalent *max* form. The excess on $a^{i\ *}_{i,\min}$ and $a^{i\ *}_{i,\max}$ relative to $a^{\downarrow}_{i,\min}$ and $a^{\downarrow}_{i,\max}$ represents the amount of water that lake i is not allowed to eject, therefore the same quantity must not be permitted to the upstream lake.

$$a^{i\ *}_{\uparrow,\min} = (v_i + a^{i}_{\uparrow})^{*}_{\min} - v_i\ ;$$
$$IF\ a^{i\ *}_{i,\min} < a^{\downarrow}_{i,\min}\ THEN\ a^{i}_{\uparrow,\min} = a^{i}_{\uparrow,\min} - a^{i\ *}_{i,\min};\ a^{i}_{i,\min} = a^{\downarrow}_{i,\min} \quad (27)$$
$$ELSE\ a^{i}_{i,\min} = a^{i\ *}_{i,\min};\ a^{i}_{\uparrow,\min} = a^{i\ *}_{\uparrow,\min}$$

At step 3, instead of the structure from Figure 12a) one could use the structure in Figure 12b), employing the same technique from step 2. The value most probably close to $u_{\uparrow}$ is calculated as:

$$a^{\uparrow}_{\uparrow,avg} = \frac{a^{\uparrow}_{\uparrow,\min} + a^{\uparrow}_{\uparrow,\max}}{2} \quad (28)$$

Using the diminishing constructions described here, the number of necessary fuzzy logic rules is reduced to $2\cdot5^2+2\cdot5\cdot9 + 5\cdot9 = 185$.

Note that for i=6, in all fuzzy inferences above (where applicable), the quantity v_i must always be taken with a minus, since v_6 is an ouput flow.

9. TESTS AND RESULTS

All simulations were carried out using the lake system presented in Figure 1. In order to make the results reproducible, the lake parameters are listed in Table 1.

Table 1. The parameters of the lakes used in simulation

lake	S_0	r_i	v_i^0	u_i^0	$g_i(h_i)$	p_i
1	10	12	0.2	0.2	1	*1*
2	25	8	0.2	0.2	2	*1*
3	30	15	0.1	0.5	1	*3*
4	22	7	0.15	0.15	1.4	*1*
5	25	12	0.15	0.3	1	*3*
6	40	12	0.4	1.2	1.5	*7*

The previous algorithms have been tested separately with the same set of disturbances. The disturbances were selected offline to resemble a period of high precipitations (v increases with 30%-40% for 200 samples) followed by a steady period (v=0% for 200 samples) and then by a period of relative drought (v decreases with 10% for another 200 samples). The variation of disturbance v_6 is considered opposite to the others, since v_6 is an output flow. The simulations were initialized with zero error (x_i=0) and were carried out until steady state was reached for each lake.

The results are displayed in Table 2, including relative peak error in water level and also information regarding the duration of the computation process. The highest peak errors for each control method are bolded. The steady state error was under 2% for each method. The fuzzy rules found for control are not displayed in this paper due to lack of space. In general, the all lakes are controlled using the same set of logic rules, except for lake 6, which has different rules.

Table 2. The results of the tests for each control strategy

	Peak errors [%]						*GA exec.*
Method \ lake	1	2	3	4	5	6	*time*
Independent	0.9	2	**6.5**	2.3	3.7	5.1	*3min*
Implicit coop. 1	4.8	4.7	**6**	4.2	4.7	2.6	*8h 21min*
Implicit coop. 2	4.8	4.7	**6.3**	5.9	4.4	2.7	*20h 9min*
Coordinated coop. 1	3	2.8	**5.1**	2.6	4.1	3.7	*26h 32min*
Coordinated coop. 2	2.5	3	**5**	3	4.4	4.4	*47h 34 min*
Explicit coop. 1	0.2	0.4	0.3	2	**1.5**	1.3	*131h 5 min*
Explicit coop. 2	1	2.1	2	**2.5**	2.4	2.4	*292h*

As expected, the independent controllers provide the lowest control performance (the peak error of water level of Lake 6 – which has the highest priority – is about 5%, and not the largest one, see Table 2).

Somewhat better results are achieved when using the implicit cooperative control strategy: the peak error for Lake 6 is under 3%, and the highest peak error – for Lake 3 – is lower than the corresponding error for independent control, see Table 2). Both implicit control strategies provide similar performance.

Even better results can be achieved using the coordinated control approach: the largest peak error for both methods is a little over 5%, and the peak error of lake 6 is a bit larger than for previous methods, but the overall performance is improved. Both cooperative control methods give similar performance, but computation process takes longer than for implicit cooperative control.

Best performance is observed – as expected – at the explicit cooperative control strategy. Although the computation process takes a very large amount of time, the results show that the efforts are not in vain. The highest peak error in water level is less than 2% for Method 1 and less than 2.5% for Method 2 (see Table 2).

Also note that explicit cooperative control – Method 1 achieves better performance than Method 2. This must be because in Method 2, the exact intention of the upstream controllers is not known; only some guidelines are provided by means of the lower and upper limits. Although the distance between the two limits should decrease when iterating step 2, it was observed that error is introduced in the system at step 3, where a very coarse approximation is made regarding the upstream control action, by calculating $a^{\uparrow}_{\uparrow,avg}$.

10. CONCLUSIONS

The system was tested under very hard conditions. Generally, the control outputs were found to have relatively few variations, thus reducing the actuator stresses. However, this was not programmed explicitly. The results show that performance and rule computation time generally increase with method complexity.

It is interesting to note that errors under 2% are irrelevant as far as the hypothetical supervisor of the system is concerned: 2% is the upper limit of the zero – error fuzzy level, as seen from the level membership function in Figure 5. This is the reason for which the steady state error of 2% can also be considered zero in fuzzy terms. When the error is zero (i.e. 2% or less), a zero disturbance generally yields zero control action (according to the control rules), therefore a steady state error of 2% or less cannot be rejected.

New control methods can be created by adding other variables in the decision context, such as constraints on the control variable u_i, predictions on the global disturbance value (given by an operator), or even past values of the control actions of neighbouring lakes (considering the slow dynamics of the lake system).

The fuzzy logic control algorithms (that are executed online) are simple and their execution durations are short. This makes them applicable to microcontrollers fulfilling the real-time constraints. Also, the volumes of data interchanged by controllers are small and the required transmission band is feasible for practical implementation.

ACKNOWLEDGEMENT

This paper was supported by the project "Doctoral studies in engineering sciences for developing the knowledge based society-SIDOC" contract no. POSDRU/88/1.5/S/60078, project co-funded from European Social Fund through Sectorial Operational Program Human Resources 2007-2013.

REFERENCES

C. Buiu, I. Dumitrache, M. Zainea, Design And Optimization Of A Fuzzy Logic Controller For A Simulated Autonomous Robot, in: *CEAI*, Vol 5, No 3,4, pages 55-63, 2003.

Y. C. Chiou, L. W. Lan, Genetic fuzzy logic controller: an iterative evolution algorithm with new encoding method, in: *Fuzzy Sets and Systems* 152, 2005, pp.617-635.

H. L. Choi, L Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation, in: *IEEE transactions on robotics*, vol. 25, no. 4, august 2009, pp. 912-926

A. Gegov, N. Gobalakrishnan, Advanced inference in fuzzy systems by rule base compression. Mathware and Soft Computing, 14 (7), 2007, pp. 201-216. ISSN 1134-5632.

M. Hakimi-Asiabar, S. H. Ghodsypour, R. Kerachian, Deriving operations policies for multi-objective reservoir systems: application of self-learning genetic algorithm, in: *Applied Soft Computing, Elsevier*, 2009.

F. Hoffmann and G. Pfister, Automatic design of hierarchical fuzzy controllers using genetic algorithms, *Proceedings of The European Congress on Fuzzy and Intelligent Technologies EUFIT'*94, 1994.

T. P. Hong, J. B. Chen, Processing individual fuzzy attributes for fuzzy rule induction, in: *Fuzzy Sets and Systems* 112, Elsevier (2000) pp. 127-140.

J. How, H. L. Choi, A. Undurti, J. Redding, An intelligent cooperative control architecture, *AIAA*, 2009

B. Innocenti, B. Lopez, J. Salvi, A multi-agent architecture with cooperative fuzzy control for mobile robot, *Robotics and Automation Systems* 55, 2007, Spain.

M. G. Joo and J. S. Lee. A class of hierarchical fuzzy systems with constraints on the fuzzy rules, in: *IEEE transactions on fuzzy systems*, vol. 13, no. 2, 2005, pp. 194-203.

M. E. Khatir, E. J. Davison, Cooperative control of large systems, in: *A Post-Workshop Volume* 2003 *Block Island Workshop on Cooperative Control*, 2003, pp. 121-138.

E. Klavins, R. M. Murray, Distributed algorithms for cooperative control, *Sensors and Actuators Networks*, Published by IEEE CS and IEEE ComSoc, 2004.

L.T. Koczy, Size reduction by interpolation in fuzzy rule bases, in: *IEEE Transactions on Systems, Man, and Cybernetics*, Part B: Cybernetics, 1997, vol. 27, issue 1, pp. 14-25,

V. Korotkich, On a general framework to study cooperative system, in: *Cooperative Control and Optimization*, Robert Murphy and Panos M. Pardalor (Eds.), Kluwer Academic Publisher, 2002, pp. 121-140

T. Leţia, O. Cuibus, I. Pop, *Implicit cooperative control achieved by generic genetic fuzzy logic, in: Proceedings* of the IEEE International Conference on Automation, Quality and Testing, Robotics, 2010, vol. 1, pp 1-6

N. Nezam, I. Dumitrache, *Mamdani, Sugeno Fuzzy Systems And Control The Output Flow Of An Equalization Basin*, in: *CEAI*, Vol. 4, No. 1, pp. 27-32, 2002.

J. Shamma (Editor) *Cooperative control of distributed multi-agent systems*. Wiley, 2007, ISBN: 978-0-470-06031-5

A, Waldock, D. Nicholson, A. Rogers, Cooperative control using the max-sum algorithm. in: *Second International Workshop on Agent Technology for Sensor Networks*, Estoril, Portugal, 2008.