AMEBA – STRUCTURAL EVOLUTIONARY OPTIMIZATION: METHOD AND TOOLBOX DEVELOPMENT

Marko Corn[(a)], Maja Atanasijević-Kunc[(b)]

[(a)] Ameba d.o.o., Logatec, Slovenia
[(b)] University of Ljubljana, Faculty of Electrical Engineering, Ljubljana, Slovenia

[(a)]marko.corn@ ameba.si, [(a)]maja.atanasijevic@fe.uni-lj.si

ABSTRACT

Evolution algorithms are optimization methods that mimic a process of a natural evolution. Their stochastic properties result in a huge advantage over other optimization methods especially when solving complex optimization problems. In the paper Agent Modelled Evolutionary Based Algorithm (AMEBA) is first presented which was developed and implemented in MATLAB as a Toolbox. AMEBA algorithm has several advantages over other evolutionary algorithms and in this article the advantage of custom designed initial solution is presented. Custom designed initial solution is a solution of the problem that is built on the base of the knowledge of the system and represents a solution which the AMEBA algorithm will try to improve. This capability is presented with the example of an evolvement of multivariable controller for the pressure – level system that represents non-linear, multivariable system which is very stiff, with the property of weak inherent coupling. The AMEBA algorithm significantly improved the initial controller solution which shows that classical controller structures can also be automatically altered to increase quality of the solution.

Keywords: evolutionary computation, ameba, pressure-level pilot plant, multivariable control design

1. INTRODUCTION

Design of controllers for complex dynamic systems is usually done with the use of system mathematical model that enables the usage of optimization with which the controllers are optimized. Several types of optimization method can be used to design a proper control system regarding simpler processes. Regarding more complex systems as are for example multivariable, non-linear, time-variant etc. a group of evolutionary computation optimization methods proved to be very efficient (Logar, Dovžan, and Škrjanc 2011; Tomažič et al. 2013).

Evolutional computation algorithms are optimization methods that mimic process of the natural evolution. In general the evolutionary algorithms can be divided into two major groups: parametrical and structural algorithms.

Parametrical algorithms evolve parameters while structural algorithms evolve structures or mapping functions. For example if we would have to design a controller for the dynamic system, parametric algorithm would demand to define parameters of the chosen controller structure (very frequently a PID controller is used). In contrast to parametrical algorithms structural algorithms do not require predefined form of the controller as they can evolve the whole controller through the evolution process. The most popular parametrical algorithms are genetic algorithms (GA) (Atanasijević-Kunc, Belič, and Karba 2006; David Goldberg 1989), evolutionary strategies (ES) (Beyer 2010), differential evolution (DE) (Storn and Price 1997) and others (Brownlee 2011). Most established structural algorithms are genetic programing (GP) that have multiple implementations from the tree based implementation (Koza 1992) to the grammatically based implementation (Whigham 1992) and the evolutionary programming that is directed into the evolvement of finite state machines (Fogel, Owens, and Walsh 1966). New method that supplements the structural group of evolutionary algorithms is Agent Modelled Evolutionary Based Algorithm (AMEBA) (Corn and Atanasijević-Kunc 2011; Corn, Černe, and Atanasijević-Kunc 2012; Corn and Černe 2012). An important property of AMEBA algorithm is that it can include external knowledge in the form of a solution which has some relative advantages while optimal result is still under investigation.

In industrial applications the most widely used controller is PID controller (Veselý and Rosinová 2011) as it is simple to integrate and its control capabilities satisfy technological needs of many processes. This solutions are frequently not optimized which leaves room for the upgrades. AMEBA algorithm can include their current solution and try to improve it by changing the structure of the controller as is demonstrated in this paper.

The paper is organized in the following way. In the next section AMEBA algorithm is presented together with

corresponding toolbox developed in Matlab. In the third section a multivariable pressure-level dynamic system is presented that was used as a testing example. In the fourth section the current solution is presented in a form of multivariable PI controller and an upgraded controller developed by the AMEBA algorithm is described. At the end the conclusions and some ideas for the future work are given.

2. AGENT MODELLED EVOLUTIONARY BASED ALGORITHM - AMEBA

All evolutionary based approaches mimic the process of natural evolution. Natural evolution can be divided in three basic elements: organism, environment and reproduction of organisms. Analogue to the natural evolution, evolutionary computation methods have entity, environment and reproductions of entities.

2.1. Entity - Agent

In AMEBA algorithm entity is represented as an agent. Agent consists of nodes and connections (Figure 1). Each node represents certain mapping function that transforms input data into output data. Nodes are receiving input information from and sending output information to other nodes via connections.

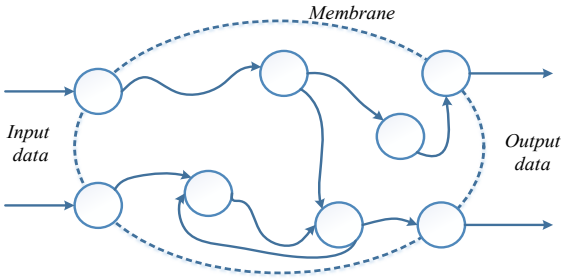


Figure 1. Representation of agent which consists of nodes and connections

In Figure 2 the structure of general node is presented. It consists of nodes inputs (u_1, u_2, ... u_n), nodes mapping function f, parameter of node's expression e, and nodes output y.

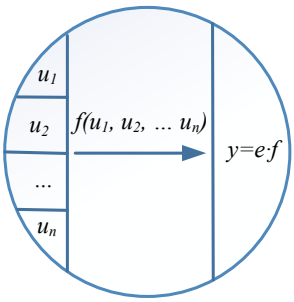


Figure 2: Structure of general node

Main task of the node is transformation of the input signal through mapping function to the output as described in equation (1).

$$y = e \cdot f(u_1, u_2, \dots, u_n) \tag{1}$$

Parameter of expression determines the influence of a node on other connected nodes. Expression parameter can reduce or amplify nodes influence on other nodes depending on the optimization process. Expression parameters initial values are randomly generated and are changing during optimization process. Expression parameter have similar role that is the expression of gens in DNA chain.

Different mapping functions determine different types of nodes (see Table 1) like basic arithmetic, logic or other functions.

Table 1: Nodes of AMEBA algorithm

Function
Input, Output
$y(k) = u_1(k)$
Constant
$y(k) = e \cdot a$
Amplify
$y(k) = e \cdot a \cdot u_1(k)$
Exponent base
$y(k) = e \cdot \lvert u_1(k) \rvert^a$
Exponent index
$y(k) = e \cdot a^{u_1(k)}$
Logarithmic
$y(k) = e \cdot log_a(u_1(k))$
Addition
$y(k) = e \cdot \sum_{i=1}^{n} a_i \cdot u_i(k)$
Multiplication
$y(k) = e \cdot \prod_{i=1}^{n} u_i(k)$
Division
$y(k) = e \cdot \frac{u_1(k)}{\prod_{i=2}^{n} u_i}$
Delay
$y(k) = e \cdot \begin{cases} u_0 & k < n \\ u_1(k-n) & k \geq n \end{cases}$
Integral
$e \cdot \begin{cases} y_0 & k = 0 \\ (k-1) + k_I \cdot u_1(k) & k > 0 \end{cases}$
Derivative
$e \cdot \begin{cases} 0 & k = 0 \\ k_d \cdot (u_1(k) - u_1(k-1)) & k > 0 \end{cases}$
High pass filter
$e \cdot \begin{cases} 0 & k = 0 \\ y(k-1) + \alpha(u_1(k) - y(k-1)) & k > 0 \end{cases}$

Low pass filter
$e \cdot \begin{cases} 0 & k = 0 \\ \alpha \cdot y(k-1) + \alpha(u_1(\mathrm{k}) - u_1(\mathrm{k}-1)) & k > 0 \end{cases}$

Special property of AMEBA algorithm is possibility of a definition of nodes with the dynamic mapping function like time delay, integral, derivative and filters. With the usage of dynamic nodes a design of feedback loops is possible and by this the formation of dynamic systems.

2.2. Environment

Natural environment provides a habitat for organisms and the mechanism of selection that determinate which organisms will be reproduced. In artificial evolution natural environment is replaced by simulation environment in which artificial entities are "living" and the selection mechanism is replaced by artificial selection mechanism based on an appropriate fitness function. Fitness function value quantify entities success at solving problem.

In AMEBA algorithm two mechanisms of selection are implemented: the survival of the best and the tournament selection. The survival of the best is a mechanism in which only the best portion of agents in a population are given a chance to reproduce. The tournament selection is a mechanism where two randomly selected agents compete and the winner gets an opportunity to reproduce. These two mechanisms were implemented because they have different effect on population diversity. Selection of the best mechanism strongly reduces diversity of the population as only the best agents survive in contrast to a tournament selection mechanism where also weaker agents can survive.

2.3. Reproduction

With the selection process a group of agents are selected that will be able to reproduce into a new population. In general there are two groups of reproduction mechanisms: mutations and crossovers. Mutation is a reproduction mechanism where an agent replicate itself and during this process slightly change one of its features. Crossover is a reproduction mechanism where two agents combine and reproduce an offspring which possess certain features of both parent-agents. In AMEBA algorithm several reproduction mechanisms are implemented (see Table 2).

Elite division enables cloning of the agent, which means that agent reproduce an exact copy of itself for the new population. By elitism the best solution is always kept trough out the whole optimization process.

Parameter change mutation mechanism reproduces an offspring from parent agent by changing value of nodes parameter if node possesses one. Change can be made from within the node by the equation (2) or with the influence of other nodes regarding the equation (3).

Table 2: Reproduction mechanism of AMEBA algorithm

Group	Reproduction
Mutations	Elite division
	Parameter change
	Add node
	Remove node
	Switch connection at source
	Switch connections at target
Crossovers	Combination of agents parameters
	Combination of agents nodes and connections

$$a' = \begin{cases} a + r \\ r \\ 1 \\ \frac{a}{-a} \end{cases} \tag{2}$$

$$a' = \begin{cases} b \\ b\ (+, -, *, /)\ c \end{cases} \tag{3}$$

Parameter change can be done in four ways. To parameter a can be added random value r, it can be replaced by random value r, it can be transform to its inverse value or it can change its sign.

Parameter change with the influence of other nodes can be done in five ways. It can be switched with parameter from randomly selected node b, or it can be calculated as sum, difference, product or quotient of two parameters b and c from two randomly selected nodes.

Add node reproduction mechanism adds a new node to agent. First the random connection is selected than the random node is generated and inserted in the place of the selected connection. If the new node have free inputs that must be connected to the network they are randomly connected to other nodes in network. In general it is not needed that node is meaningfully connected to the network it can be unconnected which represents a blind node.

Remove node reproduction mechanism removes randomly selected node from parent agent. First the node for removal is selected from the agent than all its input connections are removed. In the next step the output connections are unpinned from the node and randomly connected to some other nodes and then the selected node is removed.

Switch nodes connections reproduction mechanism switches either a source or target ends of two randomly selected connections in the network.

Combination of agents' parameters is a crossing reproduction mechanism and it is very similar to parameter change mechanism from mutations with the difference that nodes for reproduction can be from other agents in the population.

Combination of agent's nodes and connections is a crossing mechanism that combines two parent agents

into new offspring. First randomly selected group of connected nodes is selected and removed from the first parent. Then the same process is done on the second parent. Finally the removed nodes from the first parent are inserted into the second parent and randomly reconnected to the rest of the nodes network. This mechanism is similar to a reproduction mechanism of combining decision trees in genetic programming method with the difference that is made for the graph type of structure.

2.4. Toolbox development

AMEBA algorithm is being developed also as software package with the user friendly graphical interface. The core development is being built in Java programming environment that can be used also with Matlab that have a very strong support in simulation of dynamic systems via Simulink toolbox. Graphical interface is also developed in Matlab for the easier use of AMEBA algorithm.

Toolbox enables settings of the simulation environment with the inclusion of Simulink model as it is shown in Figure 3.

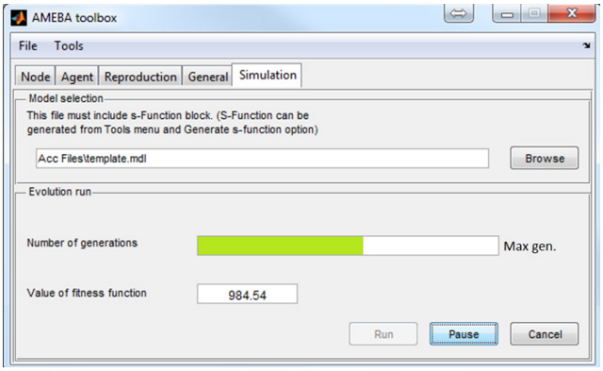


Figure 3: Settings of simulation environment

The agent of AMEBA algorithm is implemented as S-function so it can be included into model as a standard block. Toolbox enables control and monitoring of the optimization process where it displays current generation number and value of the fitness function of the best agent.

Toolbox enables settings of population properties like size of population, size of reproductive population that determines how many best agents will be given opportunity to reproduce, number of elite agents, and other settings that determine the end of optimization process like maximum number of generations and minimum change in fitness function value (Figure 4).

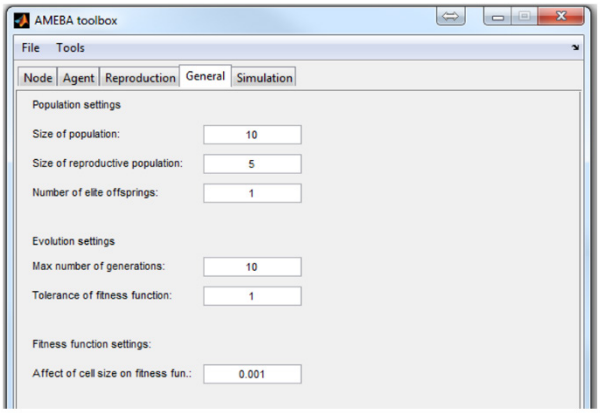


Figure 4: General setting

Number of inputs and outputs of an agent can be set together with the maximum number of nodes that can be generated at the agent's creation (Figure 5).

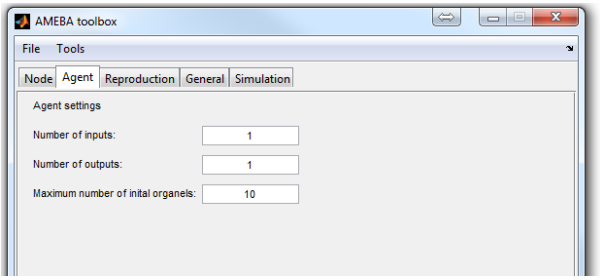


Figure 5: Agent settings

Different types of nodes can be selected from which the algorithm will chose and build agents. Each node has its own settings that determine initial value of the nodes parameter and steepness of change in case node mutates (Figure 6).

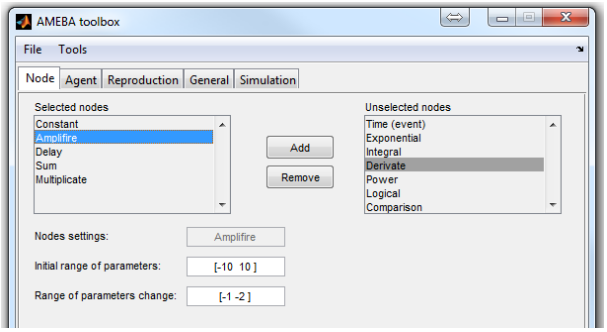


Figure 6: Node settings

Reproduction mechanisms can be set with their parameter of probability. As agents are evaluated and selected for reproduction the reproduction mechanism is randomly selected and the probability parameter determines their possibility of being selected (Figure 7).

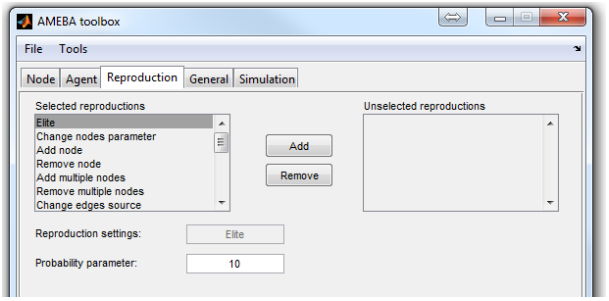


Figure 7: Reproduction settings

Additional functionalities enable better usability of the method such as saving and importing of all setting into file for later use. With this option also the initial population can be imported which enables to inclusion of certain knowledge of the solution into the optimization problem. It is also possible to convert agent into mathematical equation to observe its structure. It can also generate Matlab S-function file for the easier implementation with Simulink tool (Figure 8).

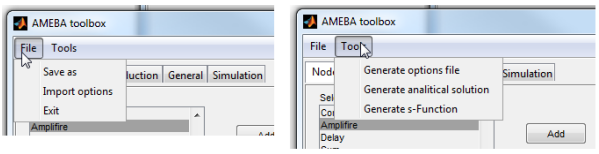


Figure 8: Additions functionalities of Toolbox

3. PRESSURE-LEVEL PILOT PLANT

Optimization is of great importance in many scientific fields where control systems' design is no exception. For testing and comparison purposes we have decided to use mathematical models of pressure–level pilot plant as presented in Figure 9 (Atanasijević-Kunc and Karba 2006; Atanasijević-Kunc, M., Belič, A., and Karba 2007).

Figure 9: Pressure-level pilot plant

Schematically it is illustrated also in Figure 10.

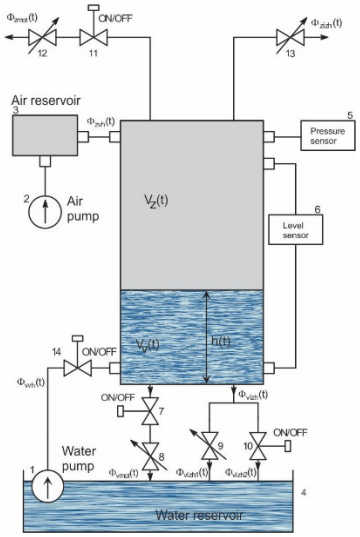


Figure 10: Schematic presentation of pressure-level pilot plant

The central part of the device is a closed tank where air pressure and water level can be controlled through two pumps (1 and 2 in Figure 9). They represent actuators of the system with an input voltage range of 0-10V. The air reservoir (3) increases the time constant of the pressure loop and equalizes pulses in the air flow. The water flow recirculates in the system and therefore there is no need for any external water input. Both outputs of the sensors are also within the 0-10V range. Normal working conditions of the system can be disturbed by changing set-points of corresponding valves, but there is no possibility for direct measurement of these signals.

From the given description it is obvious that the process can be represented by a block diagram as shown in Figure 3 where the following notation is used:

$u_1(t)$	voltage input to the air pump
$u_2(t)$	voltage input to the water pump
$\Phi_{zvh}(t)$	air flow to the closed tank
$\Phi_{vvh}(t)$	water flow to the closed tank
$p_z(t)$	air pressure in the tank
$h(t)$	water level in the tank
$y_1(t)$	voltage output from pressure sensor
$y_2(t)$	voltage output from level sensor

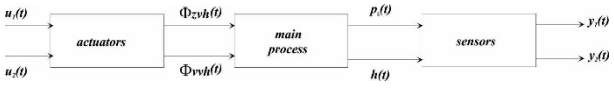


Figure 11: Block diagram of plant components

3.1. Nonlinear model

Model description of this system can be presented regarding the components indicated in Figure 11.

Actuators can be described with the following equations (Atanasijević-Kunc, M., Karba, R., and Zupančič 1997) for the air pump:

$$T_{a11}T_{a12}\ddot{\Phi}_{zvh}(t)+\left[T_{a11}+T_{a12}\right]\dot{\Phi}_{zvh}(t)+\Phi_{zvh}(t)=K_{a1}u_1^2(t) \quad (4)$$

and for the water pump:

$$T_{a211}\dot{\Phi}_{vvh}(t)+\Phi_{vvh}(t)=K_{a2}u_2^5(t) \quad (5)$$

Main process equilibrium equations are:

$$\Phi_{zvh}(t)-\Phi_{zizh}(t)=\dot{m}_z(t) \quad (6)$$

where:

$$\Phi_{zizh}(t)=K_{zv}\sqrt{p_z(t)-p_0} \quad (7)$$

Taking into account that air in the reservoir is an ideal gas, specific gas constant is given with:

$$r_z=\frac{p_zV_z}{m_zT_z}=\frac{p_z}{\rho_zT_z}=\frac{10^5N/m^2}{1.3kgm^3293^o} \quad (8)$$

so:

$$m_z(t)=\frac{p_z(t)V_z(t)}{r_zT_z(t)}=\frac{p_z(t)}{r_zT_z(t)}S\left[H_0-h(t)\right] \quad (9)$$

and

$$\dot{m}_z(t)=\frac{S}{r_zT_z}\left[\left(H_0-h(t)\right)\dot{p}_z(t)-p_z(t)\dot{h}(t)\right] \quad (10)$$

The first equilibrium equation of the main process can be presented in the following form:

$$\dot{p}_z(t)=X_1\frac{K_B}{H_0-h(t)}\Phi_{zvh}(t)-X_2\frac{K_BK_{zv}}{H_0-h(t)}\sqrt{p_z(t)-p_0}+$$
$$+X_3\frac{1}{H_0-h(t)}p_z(t)\dot{h}(t) \quad (11)$$

Equilibrium equation of the main process for the lower part can be described with:

$$\Phi_{vvh}(t)-\Phi_{vizh}(t)=\dot{m}_v(t) \quad (12)$$

where:

$$\Phi_{vizh}(t)=\sqrt{K_1h(t)+K_2\left(p_z(t)-p_0\right)} \quad (13)$$

Taking into account that water is not compressible it can be written:

$$m_z(t)=\rho_vSh(t) \quad (14)$$

and the second equilibrium equation of the main process can be presented in the following form:

$$\dot{h}(t)=\frac{1}{C_2}\Phi_{vvh}(t)-\frac{1}{C_2}\sqrt{\mathrm{K}_1h(t)+K_2\left(p_z(t)-p_0\right)} \quad (15)$$

where the following notation was introduced:

$$C_2=\rho_vS \quad (16)$$

Both **sensors** are linear and can be described with:

$$y_1(t)=K_{s1}\left(p_z(t)-p_0\right)-\bar{y}_{1off} \quad (17)$$

$$y_2(t)=K_{s2}h(t)-\bar{y}_{2off} \quad (18)$$

In all presented equations index *z* is related with the air and index *v* with the water. Parameters of this nonlinear multivariable system were estimated regarding the following conditions: valves V7, V8, V9, V11, V12 closed, valve V10 open, valve V13 open for two revolutions. Working points regarding input voltage signals were chosen as:

$$\bar{u}_1=4V$$
$$\bar{u}_2=3V$$

and so the input air and water flows at working points were estimated as:

$$\bar{\Phi}_{zvh}=22\times10^{-6}kg/s$$
$$\bar{\Phi}_{vvh}=7\times10^{-3}kg/s$$

while model parameters are for the actuators:

$$K_{a1}=\frac{\bar{\Phi}_{zvh}}{\bar{u}_1^2},\ K_{a2}=\frac{\bar{\Phi}_{vvh}}{\bar{u}_2^5},\ T_{a11}=1s,\ T_{a12}=0.8s,\ T_{a211}=1.5s,$$

for the sensors:

$$K_{s1}=0.2V/mbar,\ K_{s2}=0.4V/cm,$$

and for the main process:

$H_0=0.3m$ [reservoir height],

$S=24.10^{-4}m^2$ [reservoir cross section],

$p_0=76.10^3N/m^2$ [normal air pressure],

$X_1=X_2=X_3=0.04$,

y_{1off} = -0.0187V [air sensor off-set],

$y_{2off}=1.2524$V [water sensor off-set],

$K_B=\frac{r_zT_z}{S}=3.205\times10^7s^{-2}$,

$C_2=\rho_vS=2.3958kg/m$,

$\bar{y}_1=0.8157\text{V-}y_{1off}$ [1^{st} output signal at working point],

$\bar{y}_2=3.1133\text{V-}y_{2off}$ [2^{nd} output signal at working point],

$K_{zv}=\frac{\bar{\Phi}_{zvh}}{\sqrt{\bar{p}_z-p_0}}$ [valve V13 constant], $\bar{h}_1=0.1012m$,

$K_1=\frac{\bar{\Phi}_{vvh}^2}{\bar{h}_1}$ [valve V10 constant],

$K_2=\frac{\bar{\Phi}_{vvh}^2-K_1\bar{h}}{\bar{p}_z-p_0}$ [valve V10 constant].

3.2. Linearized model

By the following selection of system states:

$$x_1(t)=\Delta\Phi_{zvh}(t)\ \ldots\ 1^{\text{st}}\ \text{state}, \quad (19)$$

$$x_2(t)=\dot{x}_1(t)\ \ldots\ 2^{\text{nd}}\ \text{state}, \quad (20)$$

$$x_3(t)=\Delta\Phi_{vvh}(t)\ \ldots\ 3^{\text{rd}}\ \text{state}, \quad (21)$$

$$x_4(t)=\Delta p_z(t)\ \ldots\ 4^{\text{th}}\ \text{state}, \quad (22)$$

$$x_5(t)=\Delta h(t)\ \ldots\ 5^{\text{th}}\ \text{state}, \quad (23)$$

where Δ indicates the variation from chosen working points, the following linearized state-space equations are defined:

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (24)$$

$$y(t) = Cx(t) + Du(t) \quad (25)$$

where state-space matrices are given with:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -1.25 & -2.25 & 0 & 0 & 0 \\ 0 & 0 & -0.6667 & 0 & 0 \\ 5.7707\times10^{6} & 0 & 5.7429\times10^{3} & -0.1782 & -198.6386 \\ 0 & 0 & 0.4174 & -1.8918\times10^{-6} & -0.0144 \end{bmatrix} \quad (26)$$

$$B = \begin{bmatrix} 0 & 0 \\ 1.375\times10^{-5} & 0 \\ 0 & 0.0078 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (27)$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0.002 & 0 \\ 0 & 0 & 0 & 0 & 40 \end{bmatrix} \quad (28)$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (29)$$

4. CONTROL DESIGN

Control system major tasks were in the presented case focused on two important goals: to overcome system stiffness which in the same time results in better robust behavior and to overcome the property of weak inherent coupling which demands a construction of a dynamic compensator in a combination with the observer in case of the design goal of dynamic system decoupling. This is important for high quality of multivariable system operation as cross-couplings introduce disturbances in reference-tracking operation.

We have tested and compared different control design approaches. During modelling, simulation and analysis phase where also first close-loop solutions were tested certain knowledge of the system was accumulated which we tried to include into the control design phase by prescribing desired responses of the system to the reference signals as illustrated in Figure 12.

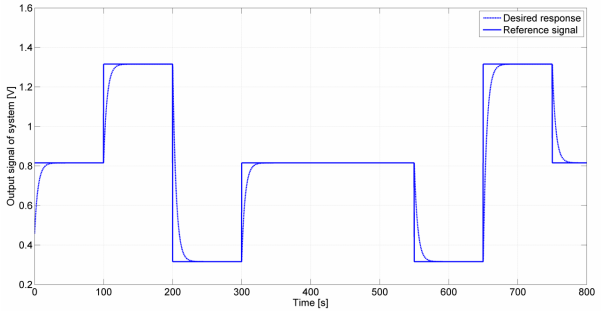


Figure 12: Example of reference signal and desired responses of the system

The velocity of transient responses was estimated regarding control signals which have to be inside the range of actuators. In case of decoupling not only cross-couplings are compensated but also relative orders of each univariable sub-system is correspondingly lower. In our case we have taken into account that approximate performance of the first order can be achieved.

Such system responses were then used in fitness function description through optimization process with which two controllers were designed. First controller is classical multivariable PI or proportional-integral controller and the second one was generated by the AMEBA algorithm.

4.1. Design of the multivariable PI controller

For the system with two input and two output signals multivariable PI controller is described with:

$$\vec{u}(t) = \boldsymbol{K_p}\vec{e}(t) + \boldsymbol{K_i}\int \vec{e}(t)dt$$

We used simplex optimization method from Matlab to calculate the eight parameters of the PI-controller as:

$$\begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = \begin{bmatrix} 8.97 & -0.037 \\ 0.71 & 2.55 \end{bmatrix} \begin{bmatrix} e_1(t) \\ e_2(t) \end{bmatrix} + \begin{bmatrix} 0.01 & -0.00004 \\ 0.00082 & 0.0029 \end{bmatrix} \int \begin{bmatrix} e_1(t) \\ e_2(t) \end{bmatrix} dt \quad (30)$$

Evaluation results are presented in Table 3.

Table 3: Results of PI controller

Error [%]	Activity of the actuators [%]
17%	34%

These results show that controller can control the system with 17% error and with the use of 35% of all capacity of actuators regarding the simulated run. Majority of error is generated by the cross-coupling effect that PI controller could not cope with. System responses are presented in Figures 13 and 14 and corresponding control signals in Figure 15.

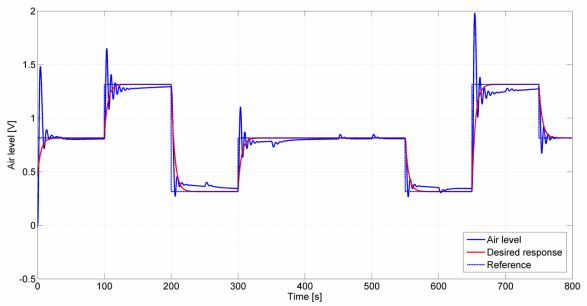


Figure 13: Response of the air-pressure to the change of reference signal

Response of the air pressure is very oscillatory which is not desired and the response of the water level shows that controller doesn't provide good reference tracking as well. Significant is also the consequence of cross-coupling effects. At the same time it is obvious that the response of the air part is much faster as that of water part.

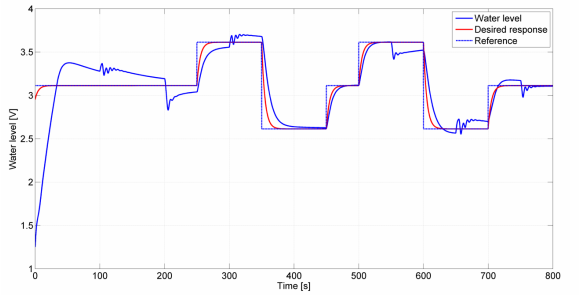


Figure 14: Response of the water level to the change of the reference signal

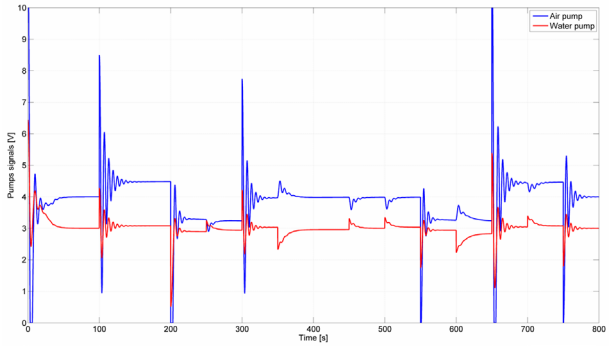


Figure 15: Activity of the air and water pumps

It is important to mention that such oscillatory responses are not desired also because of the actuators as in such cases the lifespan is expected to be reduced.

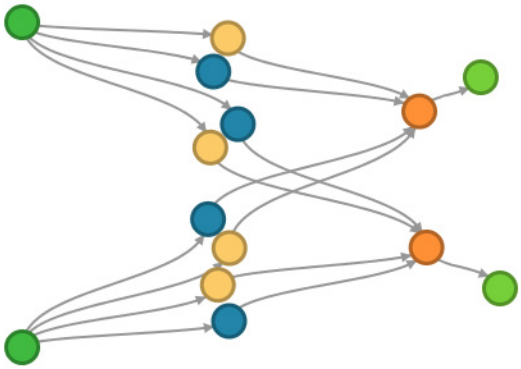
4.2. Design of the controller with the AMEBA algorithm

AMEBA algorithm has a very useful capability that it can include external knowledge of the problem into its solution. In this case we have used multivariable PI-controller that was designed through the first experiment as a starting point for the AMEBA algorithm. The agent representation of the multivariable PI-controller is presented in Figure 16.

The presented PI-controller was included as a member of the initial population in optimization process. During the optimization or evolution process there were no limitations regarding agent's structure, nor nodes' properties.

Control design results of AMEBA algorithm are presented in Table 4.

The results show an improvement of 11% in error reduction with practically no increase in actuators activities which represents significant improvement over previously presented multivariable PI-controller. The difference can be observed also in Figures 17, 18, and 19. Agent that was generated is presented in Figure 20.



Color	Node
	Input
	Output
	Amplify
	Add
	Integral

Figure 16: Representation of multivariable PI-controller as an agent of AMEBA algorithm

Table 4: AMEBA algorithm control design results

Error [%]	Activity of the acutators [%]
5,7%	36%

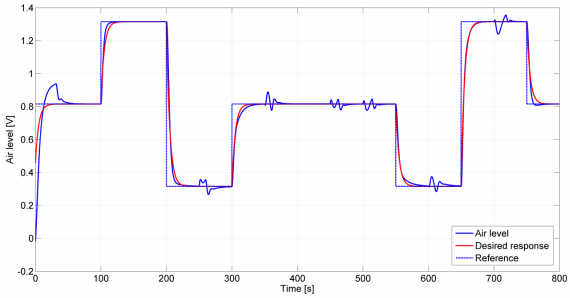


Figure 17: Response of the air pressure to the change of reference signal

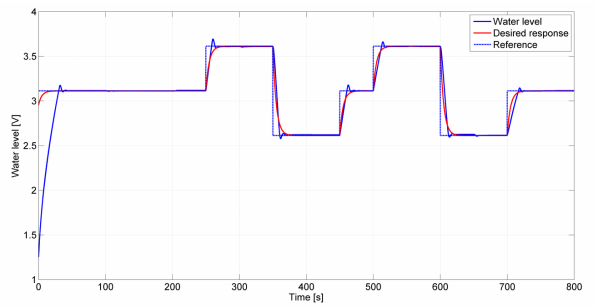


Figure 18: Response of the water level to the change of reference signal

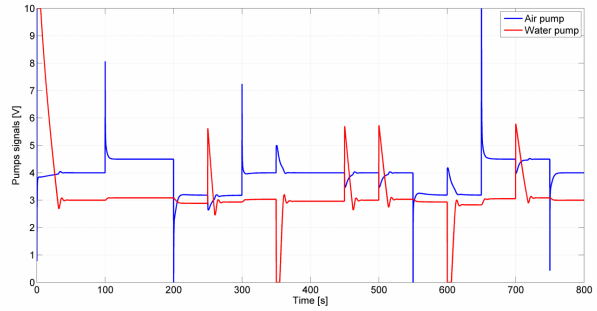
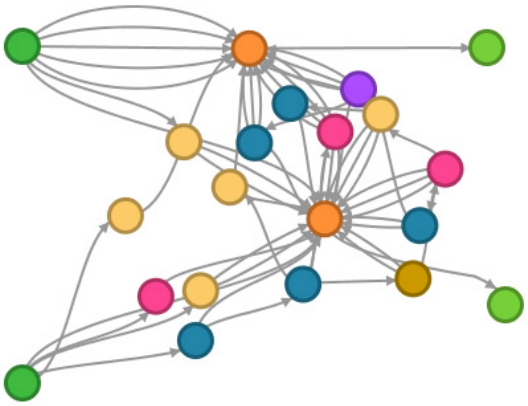


Figure 19: Activity of the air and water pumps



Color	Node
	Input
	Output
	Amplify
	Integral
	Derivative
	Delay
	Flow pass filter

Figure 20: Controller generated by AMEBA algorithm

5. CONCLUSIONS

We have presented new optimization algorithm AMEBA. Its efficacy was illustrated through the improvement of an existing multivariable PI-controller for the pressure-level pilot plant. The upgraded effects were especially evident in reduction of the cross-couplings and in reduction of the oscillations which can extend actuators' lifespan and increase control quality. Future work on AMEBA algorithm development will be focused on optimization process as we will explore the impact of various effects on the quality of the solution and on the convergence rate of optimization process like effect of size of population, effect of suppression of the agents with large number of nodes, effect of using multiple environments at once and similar, of course in comparison with other optimization approaches. Special attention will be devoted to the so called smart optimization where additional knowledge from chosen area can be taken into account to improve searching efficacy.

REFERENCES

Atanasijević-Kunc, M., Belič, A., and Karba, R., 2006. Optimal Multivariable Control Design Using Genetic Algorithms. *5th Vienna Symposium on Mathematical Modeling*, Vienna University of Technology.

Atanasijević-Kunc, M., and Karba, R. 2006. Multivariable Control Design with Expert-Aided Support, *WSEAS Trans. Syst* 5(10):2299–2306.

Atanasijević-Kunc, M., Belič, A., and Karba, R. 2007. Optimal Multivariable Control Design Using Genetic Algorithms. *EUROSIM simulation news Europe* 17:41–45.

Atanasijević-Kunc, M., Karba, R., and Zupančič, B. 1997. Multipurpose Modelling in the Evaluation of Laboratory Pilot Plant. *Simulation Practice and Theory* 5:751–76.

Beyer, H. G. 2010. *The Theory of Evolution Strategies (Natural Computing Series)*. Springer, Retrieved March 14, 2012 (http://www.amazon.com/Theory-Evolution-Strategies-Natural-Computing/dp/3642086705).

Brownlee, J. 2011. Clever Algorithms: Nature-Inspired Programming Recipes. Swinburne University in Melbourne, Australia.

Corn, M. and Atanasijević-Kunc, M., 2011. Cell Based Genetic Programming Toolbox. 20th ERK Conference, Portorož, pp.295–98.

Corn, M. and Černe, G., 2012. A Graph-Based Evolutionary Algorithm : Cell Based Genetic Programming. *Proceedings of the Fifth International Conference on Bioinspired Optimization Methods and their Applications, BIOMA,* Kranjska gora, pp. 163–72.

Corn, M., Černe, G., and Atanasijević-Kunc, M.. 2012. Balance Group Model with Smart Grid Elements. *7th Vienna Conference on Mathematical Modeling: MathMod 2012*. Vienna, February 15.-18.2012.: Vienna University of Technology.

Goldberg, D., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st ed. Massachusetts: Addison Wesley.

Fogel, L. J., Owens, A. J., and Walsh, M. J., 1966. *Artificial Intelligence through Simulated Evolution*. John Wiley.

Koza, J. R. 1992. *Genetic Programming On the Programming of Computers by Means of Natural Selection*. 6th ed. Cambridge, Massachusetts, London, England: MIT Press.

Logar, V., Dovžan, D., and Škrjanc, I., 2011. Mathematical Modeling and Experimental Validation of an Electric Arc Furnace. *ISIJ International* 51(3):382–91.

Storn, R., and Kenneth P., 1997. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization* 11:341–59.

Tomažič, S. et al. 2013. "Indoor-Environment Simulator for Control Design Purposes. *Building and Environment* 70(0):60–72. Retrieved (http://www.sciencedirect.com/science/article/pii/S0360132313002527).

Veselý, V., and Rosinová, D., 2011. Robust PSD Controller Design. *18th International Conference on Process Control*. Tatranská Lomnica, Slovakia, pp. 479–84. Retrieved (https://www.kirp.chtf.stuba.sk/pc11/data/papers/077.pdf).

Whigham, P. A. 1992. Grammatically-Based Genetic Programming. *Workshop on Genetic Programming: From Theory to Real-World Applications*. Tahoe City, California: Departent of Computer Science, University College, University of New South Vales, pp. 33–41.